Kovács Magda
Knapp Gábor • Ágoston György
Budai Attila

BEVEZETÉS A SZÁMÍTÁSTECHNIKÁBA



Dr. Kovács Magda
Knapp Gábor • Ágoston György
Budai Attila

BEVEZETÉS A SZÁMÍTÁSTECHNIKÁBA

Nyitott rendszerű képzés – távoktatás –
oktatási segédlet
Felsőoktatási Tankönyv

LSI Oktatóközpont
A Mikroelektronika Alkalmazásának
Kultúrájáért Alapítvány

Budapest, 1999

Készült az Oktatási Minisztérium támogatásával

Szerzők:

I., II. fejezet	**Dr. Kovács Magda, Budai Attila**
III., IV. fejezet	**Dr. Kovács Magda**
V. fejezet	**Knapp Gábor**
VI. fejezet	**Dr. Kovács Magda, Budai Attila**
VII. fejezet	**Ágoston György**

Lektorálta: **Dr. Szelezsán János** kandidátus

Dr. Bíró Gábor kandidátus

A rendszerezésben, technikai szerkesztésben közreműködött: **Budai Attila**

Szerkesztés, tördelés: **Kántorné Padra Anna** OFFICE-CAR Bt.

Korrektor: **Márkus Orsolya**

ISBN 963 577 270 X

Kiadó: **LSI Oktatóközpont**
Felelős vezető: **Dr. Kovács Magda**
Témafelelős: **Flier István**
LIGATURA KFT – ÁFÉSZ NYOMDA, Vác

JELÖLÉSEK MAGYARÁZATA

A könnyebb áttekinthetőség érdekében különböző jelekkel találkozik a könyvben. Az alkalmazott jelek és jelentésük:

! – Fontos kérdésekre hívja fel a figyelmet, definíció

? – Ellenőrző kérdések, megválaszolásukkal ellenőrizheti az elsajátított ismeretek szintjét

 – Súlyponti rész, figyelmesen kell olvasni

 – Számítógéppel végrehajtandó gyakorló feladat

 – Aprólékos, figyelmes áttanulmányozást, önálló munkát igénylő ismeretanyag, fontos összefüggéseket tartalmazó logikai vázlat

 – Kiegészítő ismeretek

(1)2.3 – (Al)Fejezet azonosító

TARTALOMJEGYZÉK

I. BEVEZETÉS 13
A félvezető technológia és az elektronikai rendszertechnika fejlődése. A számítógépek kialakulása 13

II. INFORMATIKAI ALAPISMERETEK 19
A. ALAPFOGALMAK 19
1. A Neumann-elvű számítógép 19
2. A rendszer fogalma 24
3. Az algoritmus és a program fogalma 25
4. Információ és adat 27
5. Információtechnológia, információs és adatfeldolgozó rendszer, informatika 29
6. Kódolás, kódrendszer, kódkészlet 31
7. A hardver, szoftver és a förmver fogalma 32

B. AZ ANALÓG ÉS DIGITÁLIS TECHNIKA 33
1. Analóg technika 34
2. Digitális technika 35
3. Jelátalakítás 35
3.1. Analóg–digitális átalakítás 35
3.2. Digitális–analóg átalakítás 36
4. Logikai áramkörök 36
4.1. Bináris logika 36
4.2. Boole-algebra 37
4.2.1. Boole-művelet 39
4.2.2. Igazságtáblázat 39
4.3. Kapuáramkörök, logikai áramkörök 41

C. DIGITÁLIS ADATÁBRÁZOLÁS A SZÁMÍTÓGÉPEKBEN 47
1. Számrendszerek 47
1.1. A tízes (decimális) számrendszer 47
1.2. A kettes (bináris) számrendszer 48
1.3. A tizenhatos (hexadecimális) számrendszer 48
1.4. Számrendszerek átalakítása 49

1.4.1. Decimális–bináris átalakítás ... 49
1.4.2. Bináris–decimális átalakítás ... 50
1.5. Műveletvégzés bináris számokkal ... 50
1.5.1. Bináris számok összeadása ... 50
1.5.2. Bináris számok kivonása ... 51
2. A számítógépes számábrázolás ... 53
2.1. Fixpontos számok ... 53
2.2. Binárisan kódolt decimális számok (BCD kód) ... 55
2.3. Lebegőpontos számok ... 57
3. Karakterek kódolása ... 58
Ellenőrző kérdések ... 60

III. A MIKROSZÁMÍTÓGÉP ... 63
1. Bevezetés ... 63
2. A mikroszámítógép felépítése ... 64
2.1. A központi egység ... 64
2.2. Perifériák ... 65
3. A mikroszámítógép működése ... 65
3.1. Órajel és gépi ciklus ... 65
3.2. Tárak, tárolók (memória) ... 66
3.3. Memóriacímzés ... 70
3.4. A gépi utasítások végrehajtásának lépései ... 73
4. A mikroszámítógép szoftvere ... 74
Ellenőrző kérdések ... 77

IV. HARDVER ALAPISMERETEK ... 79
A. MIKROPROCESSZOR ... 79
1. Bevezetés ... 79
2. A mikroprocesszor feladata ... 79
3. A mikroprocesszor funkcionális egységei ... 80
3.1. Regiszterek ... 80
3.1.1. Tipikus regiszterek ... 80
3.1.2. A regiszterek használata, az utasításvégrehajtás elemi lépései ... 83
3.2. Aritmetikai logikai egység (ALU) ... 84
3.3. Vezérlőegység (CU) ... 85
3.4. Állapotinformáció ... 86
3.4.1. A jelzőbit ... 87
3.4.2. Állapotregiszter ... 87
4. A processzor utasításkészlete, címzési eljárások ... 87

4.1. A gépi utasítás szerkezete ... 87
4.2. Címzési módszerek ... 88
4.2.1. Az abszolút címzés ... 88
4.2.2. Relatív címzés ... 89
4.2.3. Közvetlen adatcímzés ... 90
4.2.4. Veremcímzés (STACK-címzése) ... 91
4.2.4.1. Memóriaverem ... 91
4.2.4.2. Veremtár műveletek ... 91
4.2.4.3. A kaszkád verem ... 93
4.2.4.4. Mire használható a veremtár? ... 95
4.2.4.5. Egymásba illesztett – egymásba ágyazott – szubrutinok és a veremtár használata ... 98
4.2.5. Közvetett címzés ... 99
4.2.6. Indexelt címzés ... 99

B. A MIKROPROCESSZOR ALAPÚ RENDSZER ... 101
1. Memóriák csatlakoztatása a mikroprocesszorhoz ... 101
1.1. ROM (Read Only Memory) csak olvasható tár ... 101
1.2. RAM (Random Access Memory) közvetlen elérésű írható-olvasható tár ... 104
2. Adatátvitel a mikroszámítógép és a hozzá csatlakoztatott perifériális egységek között ... 106
2.1. Interfész, I/O PORT ... 107
2.2. Az adatátvitel típusai ... 109
2.3. Programozott adatátvitel (PIO mód) ... 109
2.4. Programmegszakítással (interrupt) történő adatátvitel ... 112
2.4.1. A programmegszakítás okai ... 113
2.4.2. A megszakítás kiszolgálása ... 113
2.4.3. A megszakítások típusai ... 114
2.5. Adatátvitel közvetlen memóriahozzáféréssel (DMA) ... 114
2.5.1. DMA adatátviteli eljárások típusai ... 115
2.5.2. A DMA regiszterei ... 115
2.5.3. A DMA művelet végrehajtása ... 117
2.5.4. DMA vezérlőfunkciók ... 117
2.6. Külső buszrendszer (system bus) ... 118
2.7. Soros adatátvitel ... 119
2.7.1. A soros és párhuzamos adatátvitel ... 119
2.7.2. A soros adatátvitel bitjeinek felismerése ... 120
2.7.3. Az adó és vevő szinkronizálása ... 122
2.7.4. Kommunikációs protokoll ... 122

2.7.5. Soros adatátvitel távbeszélő vonalakon ... 123
2.7.6. Szinkron és aszinkron soros adatátvitel ... 124
Ellenőrző kérdések ... 125

V. **SZOFTVER ALAPISMERETEK** ... 129
1. Bevezető ... 129
2. Az operációs rendszer ... 132
2.1. Az operációs rendszer felépítése ... 132
2.1.1. A rendszermag alsó (hardver) felülete ... 133
2.1.2. A rendszermag felső (szoftver) felülete ... 134
2.1.3. A rendszermag „magja” (röviden) ... 135
2.2. Állománykezelés ... 135
2.2.1. Elnevezések, hivatkozások ... 136
2.2.2. Az állományok egyéb jellemzői ... 137
2.2.3. Katalógusok (directory) ... 138
2.2.4. Kötetek (volume) ... 139
2.2.5. Hivatkozások ... 140
2.2.6. Közvetett elérés, keresési útvonal ... 141
2.2.7. Fájlok fizikai elhelyezése ... 142
2.3. A felhasználói felület ... 145
2.3.1. Karakteres felhasználói felület, a parancsértelmező ... 146
2.3.2. Grafikus felhasználói felületek ... 147
2.3.3. Segédprogramok, alrendszerek ... 148
3. Programkészítés ... 149
3.1. A forráskód elkészítése ... 150
3.2. Fordítás ... 150
3.3. Szerkesztés ... 151
3.4. Betöltés, dinamikus könyvtárak ... 152
3.4.1. A program összeállítás teljes folyamata ... 152
4. Röviden a hálózatokról ... 154
4.1. Mire jó a számítógép-hálózat? ... 154
4.2. A számítógépek közötti kommunikáció ... 155
4.2.1. Az összeköttetés ... 156
4.2.2. Közös nyelv (protokoll) ... 158
4.2.3. Címek ... 160
4.2.3.1. Fizikai címek ... 160
4.2.3.2. IP címek ... 161
4.2.3.3. Levelezési címek ... 162
4.3. Hálózattípusok ... 162
4.4. Munkamegosztás a hálózaton ... 163

Ellenőrző kérdések ... 165

VI. AZ UTASÍTÁSKÉSZLET ÉS FELHASZNÁLÁSA A PROGRAMOZÁSBAN ... 167
1. A mikroszámítógép modell felépítése ... 168
1.1. Regiszterkészlet ... 170
1.2. Állapotjelzők ... 172
1.3. I/O portok ... 173
2. A modellprocesszor utasításszerkezetének és készletének megtervezése ... 175
2.1. Címzési módok ... 175
2.2. A gépi utasítás szerkezete ... 178
2.3. Az utasításkészlethez tartozó utasítások kiválasztása és kódolása ... 183
3. Utasításkészlet gépi kódban, a gépi kódú utasítások végrehajtása ... 187
4. Az utasításkészlet ... 195
4.1. Aritmetikai utasítások ... 195
4.2. Logikai utasítások ... 197
4.3. Adatmozgató utasítások ... 199
4.4. Vezérlésátadó utasítások ... 202
4.5. Bitléptető és forgató utasítások ... 209
4.6. Processzorvezérlő utasítások ... 211
5. Az utasításkészlet felhasználása a programozásban ... 212
5.1. A programozási feladat ... 212
5.2. Tömbök kezelése a programokban ... 221
Ellenőrző kérdések ... 224

VII. AZ IBM PC A GYAKORLATBAN ... 227
1. Az IBM PC számítógéptípus részei ... 227
1.1. Mikroprocesszor (Central Processing Unit) ... 228
1.2. ROM ... 229
1.3. RAM ... 231
1.4. Gyorsítótár (cache) ... 232
1.5. Merevlemez (hard disk, fixed disk, winchester) ... 232
1.6. Hajlékonylemez (floppy disk) ... 233
1.7. Optikai lemezek ... 234
1.8. Hangkártya ... 236
1.9. Modem ... 236
1.10. Tápegység (Power Supply), szünetmentes tápegység (Uninterrupted Power Supply, UPS) ... 237

1.11. CMOS RAM és akkumulátor ... 237
1.12. Egyéb kapcsolók ... 238
2. A személyi számítógépekről ... 238
2.1. Rövid történet ... 238
2.2. Ma modern konfiguráció ... 240
2.3. Régebbi PC egységek ... 243
2.4. A számítástechnika piac vezető vállalatai ... 244
2.5. Hol használják a számítógépet? ... 245
2.6. Y2K ... 245
2.7. Legális, vagy illegális? ... 246
2.8. A számítógép és az ember kapcsolata ... 247
3. A Pentium mikroprocesszor felépítése ... 249
4. Sínek ... 250
5. Szabványos interfészek ... 252
5.1. Soros port ... 252
5.2. Párhuzamos port ... 253
5.3. Game port ... 254
5.4. SCSI ... 254
5.5. USB ... 254
6. Háttértárak ... 255
6.1. A merevlemez (winchester) ... 255
6.2. A hajlékonylemez (floppy disk) ... 257
6.3. Streamer ... 258
6.4. DAT ... 259
7. Perifériák ... 259
7.1. Billentyűzet ... 259
7.2. Monitor ... 263
7.3. Egér ... 267
7.4. Nyomtató ... 268
7.5. Lapolvasó (scanner) ... 270
8. Az operációs rendszer betöltése ... 270
9. A számítógép használatának első lépése: a DOS ... 275
9.1. Bevezetés ... 275
9.2. MS-DOS gyakorló feladatok ... 277
9.3. DOS I. (alapok) ... 278
9.4. DOS II. (finomságok) ... 298
9.5. Miről nem esett szó? ... 313
9.6. A DOS parancsok összefoglalása ... 315
10. Commander programok ... 318
11. Tömörítő programok ... 326
12. Víruskereső és -irtó programok ... 332
13. A Windows ... 336

14. Az Internet ... 337
14.1. Elektronikus levelezés (e-mail) ... 337
14.2. World Wide Web (WWW. „világméretű pókháló”) ... 342
14.3. Telnet ... 343
14.4. Egyéb lehetőségek ... 344
15. Néhány gondolat a programozásról ... 345
15.1. Gépi kód ... 346
15.2. Assembly ... 346
15.3. C nyelv ... 347
15.4. Pascal ... 349
15.5. A programkészítés lépései ... 349
Ellenőrző kérdések ... 350

Irodalomjegyzék ... 353

Melléklet ... 355

I. fejezet

BEVEZETÉS

A FÉLVEZETŐ TECHNOLÓGIA ÉS AZ ELEKTRONIKAI RENDSZERTECHNIKA FEJLŐDÉSE

A SZÁMÍTÓGÉPEK KIALAKULÁSA

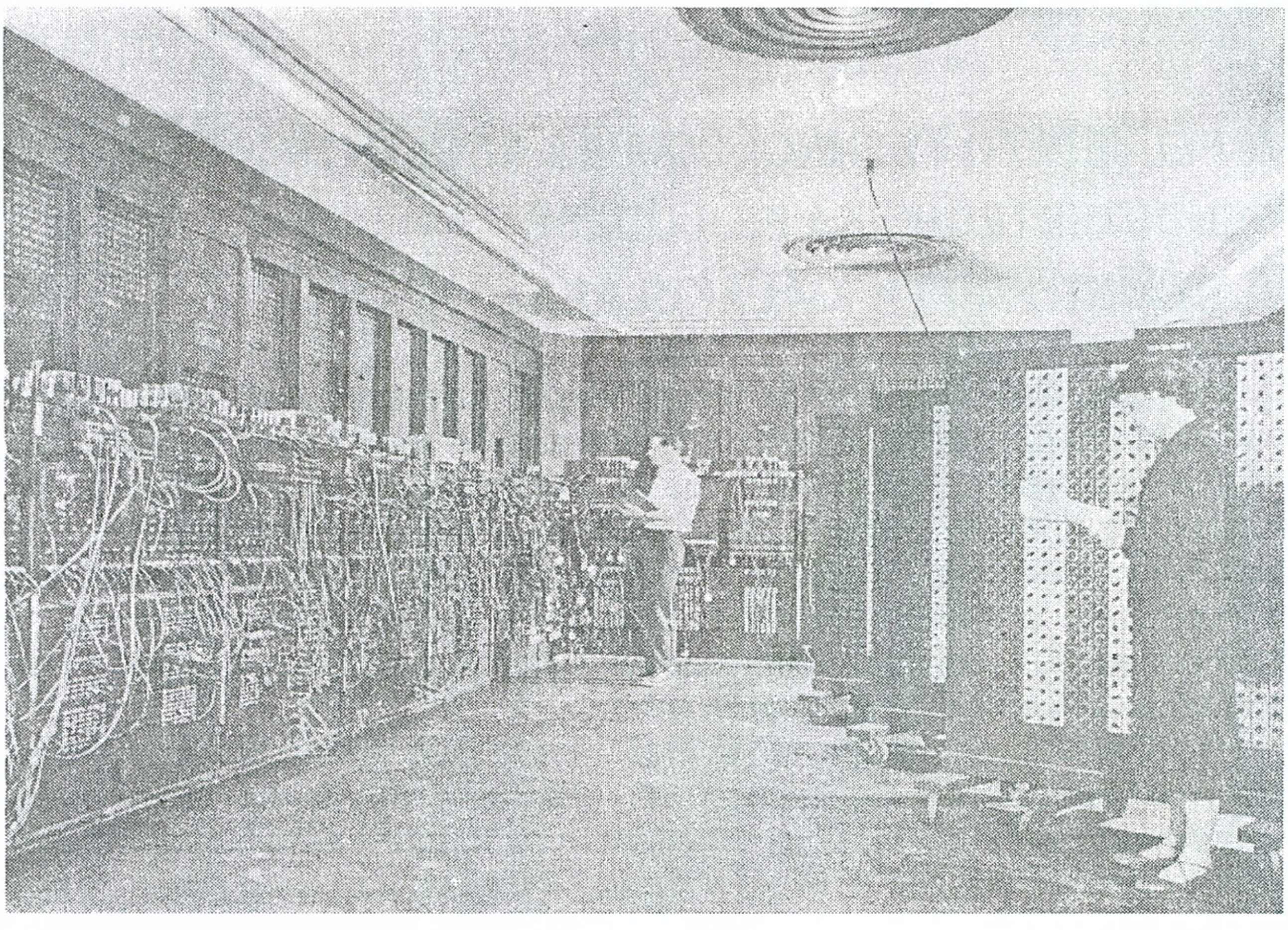
A korszerű számítógépek lényegét alkotó félvezető eszközök, integrált áramkörök fejlődésének története alig fél évszázadra nyúlik vissza a technika történetében. Ez a fejlődéstörténet azonban a 20. századi tudományok és a mérnöki teljesítmény egyik klasszikus sikertörténete. Ebben a sikerben szinte egyenlő mértékben osztoznak a fizikusok, a matematikusok, a vegyészek, a kutatómérnökök, a technológusok, továbbá más tudományágak területén dolgozó kutató-fejlesztő tudósok és szakemberek is.

A félvezető eszközök konstrukciójának, gyártási technológiájának és alkalmazásának fejlődése a II. világháborút követő időszakban minden korábbi lehetőséget felülmúlt és a fejlett ipari országokat példátlan anyagi és szellemi tőke koncentrációval az ipari társadalom-

ből, nem minden társadalmi feszültség nélkül, az ezredfordulóra az információs társadalomba vezeti át. Ez a fejlődés elképzelhetetlen a korszerű elektronika és ennek egyik csúcsterméke, a számítógép nélkül.

De hol kezdődött mindez?

A mai értelemben vett számítógépek megjelenéséig alig több mint 50 évet kell csak visszamennünk az időben. Megalkotásukhoz a nagytömegű számítások gyors elvégzésének igénye vezetett a második világháború alatti és az azt követő években. Az első elektroncsövekből felépülő számítógép az ENIAC (Electronic Numerical Integrator and Computer) volt (1. sz. kép), melynek megépítését a II. világháború nagytömegű ballisztikai számításai (tüzérségi táblázatok) tették halaszthatatlanná 1946-ban.

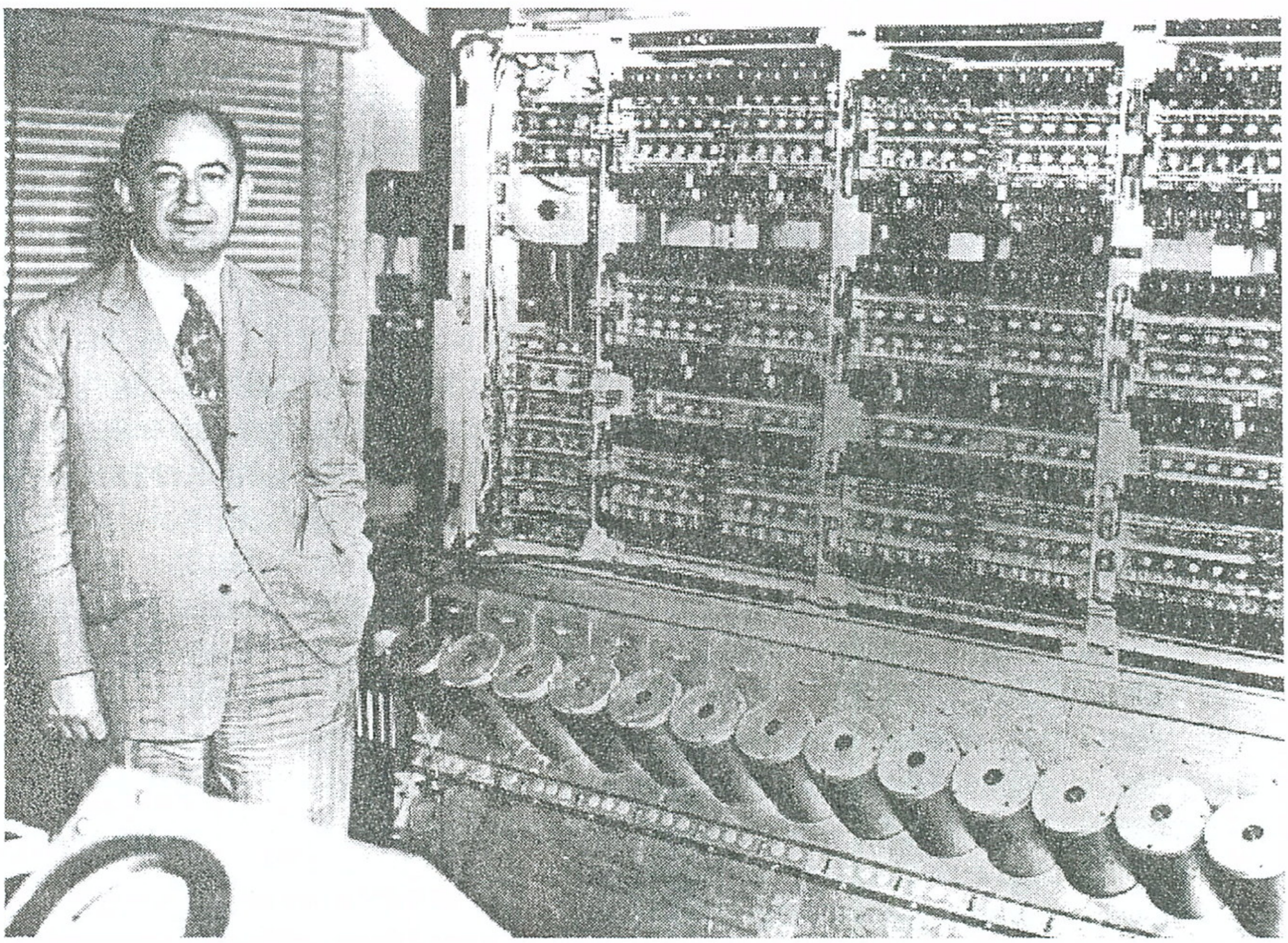


1. sz. kép
Az első elektroncsövekből felépülő számítógép az ENIAC
(USA Army foto)

Érdemes megemlíteni, hogy ez az eszköz 18.000 elektroncsövet, 10.000 kondenzátort, 70.000 ellenállást és 6000 kapcsolót tartalmazott, 30 m hosszú, 3 m széles és 1 m magas volt, 800 kW-ot fogyasztott óránként.

Az ENIAC a matematikai számításokhoz szükséges műveletsorozatot és adatokat már elektronikusan tárolta, így az aritmetikai számításokat automatikusan, emberi beavatkozás nélkül képes volt végrehajtani. Ez forradalmi változást jelentett a nagytömegű számítást igénylő feladatok megoldásában, mivel az emberi adatbevitel (például a számok bebillentyűzése) nem fékezte a műveletvégrehajtás sebességét.

Az ENIAC működtetésének tapasztalatait elemezve a magyar származású Neumann János fogalmazta meg a számítógépek felépítésével és működésével kapcsolatos elveket és követelményeket.



2. sz. kép
Neumann János

Neumann javaslatai szerint építették meg 1949-ben az EDVAC nevű (Electronic Discrete Variable Automatic Computer) számítógépet, melynek felépítése funkcionálisan már megfelelt a mai számítógépeknek. Ezt követően az események felgyorsultak.

A fejlődés első – a 60-as évek végéig tartó – szakaszában megtörtént a számítástechnika alapjainak kidolgozása, kialakultak az emberi nyelvhez közel álló, magas szintű programnyelvek, megkezdődött a számítógépek sorozatgyártása és kereskedelmi forgalmazása. Az elkövetkező években beindult a számítógépek egyre szélesebb körű alkalmazása, kialakult a számítástechnikai világpiac.

A számítástechnika gyors elterjedésének okai közül két tényezőre érdemes felhívni a figyelmet:

- a számítógépek egyre növekvő műveletvégző képessége lehetővé teszi a szellemi munka termelékenységének jelentős fokozását. Ez a piacgazdaságokban jelentős társadalmi igénnyel is találkozott, mert a szervezetek érdekeltek voltak működésük hatékonyságának növelésében (pl. a költségek csökkentésében);
- a számítógépek a konkrét feladatokhoz tartozó műveletsorozatokat képesek voltak hosszú időn keresztül tárolni és szükség esetén ismételten végrehajtani. Ezáltal a számítógépek memóriájában felhalmozott szellemi értékek mennyisége évről-évre nőtt, ami a számítógépek alkalmazási lehetőségeit is egyre inkább kiszélesítette.

A tranzisztorhatásnak 1948-ban (később Nobel-díjjal elismert) felfedezését követően csak egy évtized kellett ahhoz, hogy az egyszerű germánium tűs-tranzisztorral kezdődött rövid germánium korszak átadja helyét a szilícium egykristályból készített eszközöknek. 1959-ben jutottak el az integrált áramkör felfedezéséig, aminek van akkora jelentősége az emberi társadalom történetében, mint a tűz megszelídítése vagy a kerék felfedezése.

A félvezető eszközök atomfizikai, kristályfizikai stb. elveken alapuló konstrukciójának és az előállítási technológiának példátlanul gyors és interaktív fejlődése lehetővé tette az áramköri elemek összeintegrálását, vagyis az integrált áramkörök (IC-k) elkészítését és az integráltsági fok fokozatos növelését. Ezzel az egy piciny szilícium egykristály-darabban, a „chip"-ben elhelyezett alkatrészeknek (tranzisztoroknak, diódáknak és ellenállásoknak) a száma nagyságrendekkel növelhetővé vált. Ez a fejlődés vezetett oda, hogy többek

között nagy kapacitású, gyors és fajlagosan egyre olcsóbb tárolók és memória-áramkörök kerültek a felhasználók, a számítógép-konstruktőrök kezébe. Az alkalmazási lehetőségek bővítésére irányuló kutatások eredményeként egy amerikai diák diplomamunkájában 1969-ben megszületett az első mikroprocesszor (µP). Azóta gyorsabban írják a műszaki és információs társadalom történetét. Ma már tudjuk, hogy az elektronikus társadalomba vezető utat kikövező tranzisztort követően a mikroprocesszor a legizgalmasabb műszaki újdonság. Az első, alig több mint negyedszázada elkészült mikroprocesszortól hosszú, de sikerekben gazdag út vezetett el a ma ismert rendszerekig.

Az első mikroprocesszor előállítására akkor nyílt lehetőség, amikor egy szilíciumlapkán belül legalább húszezer kapuáramkört lehetett egyetlen monolitikus áramkörré integrálni.

Mi is az a híres, nevezetes, világformáló mikroprocesszor?

A mikro jelző a kis fizikai méretre (kis fogyasztásra, alacsony árra) utal, ami a félvezető technológiák és a rendszertechnika fejlődésének az eredménye. A processzor a digitális számítógép központi egysége. Ez irányítja a teljes számítógépet az ember által készített programok szerint. A mikroprocesszor olyan nagy integráltságú félvezető eszköz, amely egy digitális számítógép központi egységének a feladatait végzi el: dekódolja az utasításokat, vezérli a műveletek elvégzéséhez szükséges belső adatforgalmat és a csatlakozó perifériális berendezések tevékenységét. A mikroprocesszor minden logikai kifejezéssel leírható kombinációs és szekvenciális műveletsor elvégzésére alkalmas. Hogy milyen feladatot lásson el, azt a hozzá csatlakoztatott tárolókba beírt programok határozzák meg. A tárolókba írt programokat a mikroprocesszor hívja le és a rendelkezésre álló, vele összeépített eszközök közreműködésével végrehajtatja.

A különböző célokra készített mikroprocesszorok elsősorban egyes elemeknek egymáshoz való kapcsolódásában és azoknak a chipben való elhelyezkedésében térnek el egymástól. Az adatok mozgatása a mikroprocesszor és környezete között buszrendszeren (sín) keresztül történik.

Önmagában a mikroprocesszor nem használható eszköz. A gyakorlatban működő digitális rendszerek megvalósításához a mikropro-

cesszort ki kell egészíteni a típustól függően különböző funkcionális egységekkel. Az önálló működésre alkalmas rendszereket mikroszámítógépeknek nevezzük.

A mikroszámítógépek teljesítőképességének és működési sebességének növelését a mikroelektronikai gyártástechnológia fejlődésének lehetőségei ill. korlátai alapvetően meghatározzák.

Így a mikroprocesszor alkalmazási területét behatárolják a szóhosszúság, az utasításkészlet, a ciklusidő, a címezhető memóriakapacitás, a megszakításrendszer, a buszrendszer, a funkcionális egységek stb., és nem utolsósorban ezek ára. A fejlődés az utolsó évtizedben tovább gyorsult. Erre jó példa az 1997-ben kibocsátott Pentium-II, az Intel cég mikroprocesszora, mely egyetlen szilíciumlapkán elhelyezve mintegy 7,5 millió tranzisztorból épül fel.

II. fejezet

INFORMATIKAI ALAPISMERETEK

A. Alapfogalmak

1. A Neumann-elvű számítógép

Ha számítástechnikával kezdünk foglalkozni az első kérdés, amit fel kell tennünk magunknak az, hogy mi az oka a számítógépek rohamos elterjedésének az élet csaknem minden területén. Mit tudnak a számítógépek, amivel az elmúlt évtizedekben nélkülözhetetlenné váltak a számunkra?

Valóban képesek-e a számítógépek az emberhez hasonlóan gondolkodni, vagy csak nagyon gyors működésű számológépek?
E kérdés megválaszolásához vizsgáljuk meg, hogy a számítógép működése mennyiben tér el elődjeitől, a különböző elveken működő számológépektől. Nézzük meg például, hogyan számíthatjuk ki a kör területét e két eszközzel.

$T = R^2 \cdot \pi$	
SZÁMOLÓGÉP	SZÁMÍTÓGÉP
1. lépés: R bebillentyűzése 2. lépés: R bebillentyűzése 3. lépés: Szorzás műveleti jelének beadása 4. lépés: π bebillentyűzése 5. lépés: Szorzás műveleti jelének beadása 6. lépés: Az eredmény megjelenése a kijelzőn	1. A műveletsorozat megfogalmazása: • R beolvasása a számítógépbe • π beolvasása a számítógépbe • $T = R \cdot R \cdot \pi$ • az eredmény kiíratása 2. A műveletsorozat beolvasása a számítógép tárolójába 3. A műveletsorozat automatikus végrehajtása a számítógépben 4. Az eredmény kiíratása

II.1. sz. táblázat
A számológép és a számítógép működésének összehasonlítása

Látható, hogy a számológép esetében a műveleti utasításokat és adatokat kívülről, lépésenként (és emberi sebességgel) közöljük a géppel. A végrehajtást követően a műveleti sorrendet a számológép nem tárolja.

A számítógép esetében a műveletsorozatot és az adatokat is a gép memóriájában tároljuk, a műveletek végrehajtása automatikusan, emberi beavatkozás nélkül történik.

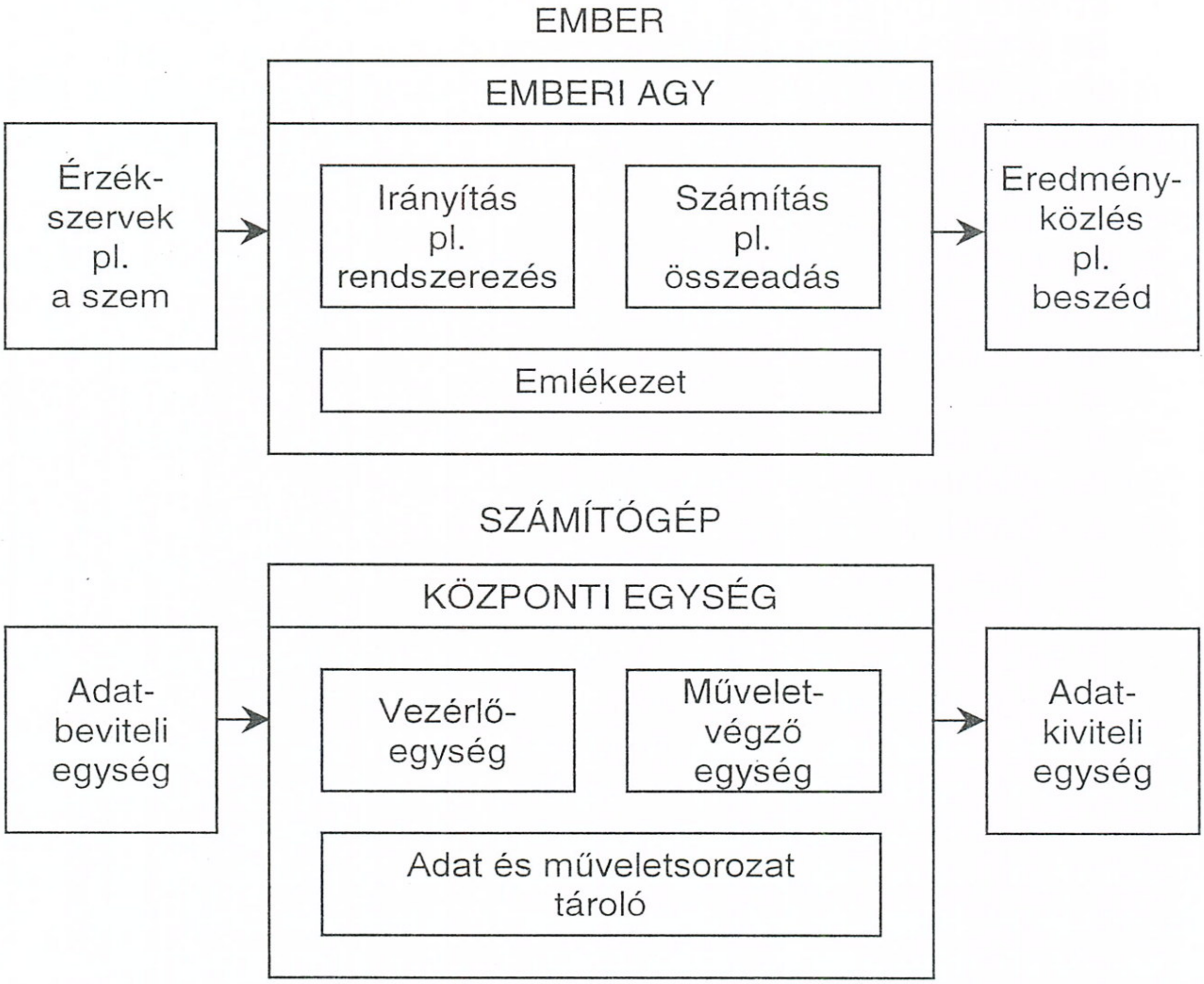
Példánkból is kitűnik, hogy a számítógép nem képes gondolkodni, viszont azokat a műveletsorozatokat, más néven programokat, melyeket az ember megfogalmaz számára (és amilyen műveletek végrehajtására felkészítették) képes tárolni és önállóan végrehajtani.

Ennyi előkészítés után megfogalmazhatjuk, hogy mit értünk számítógépen (computer, computer system):

Számítógépnek nevezzük azokat az eszközöket, melyek adatok és az ember által megfogalmazott műveletsorozatok (programok) tárolására és automatikus végrehajtására képesek. A számítógép fogalmába beleértjük a működéséhez szükséges programok összességét is.

A számítástechnika fejlődésének kiindulópontját jelentő Neumann elvek, a számítógépek legfontosabb jellemzőit rögzítik.

Érdemes megemlíteni, hogy Neumann János az emberi gondolkodás analógiái alapján alakította ki javaslatait. Ezt érzékelteti a következő ábra.



II.1. sz. ábra
Az emberi agyműködés és a számítógép funkcionális felépítésének összehasonlítása

A számítógépek felépítésének és működésének Neumann János által megfogalmazott elvei a következők:

- a tárolt program elve,
- a címezhetőség elve,
- önálló adat be/kiviteli egység, vezérlő és műveletvégrehajtó egység,
- teljesen elektronikus gép, mely a kettes (bináris) számrendszer alkalmazásával működik,
- a soros utasítás-végrehajtás elve.

A tárolt program elve azt mondja ki, hogy az elvégzendő műveleteket először be kell vinni a számítógép tárolójába, azután a gép vezérlését át kell adni ennek a műveletsornak, azaz a programnak. Ezt követően már a számítógép tárolójában lévő program irányítja a gépet:

- beolvassa az adatokat,

- elvégzi a műveleteket, ehhez tárolja a részeredményeket
- és kijelzi a végeredményt.

A címezhetőség elve azt jelenti, hogy az utasítások és az adatok ugyanolyan módon megcímzett (címmel, sorszámmal ellátott) memóriarekeszekben helyezkednek el, azaz nincsen külön hely az utasításoknak és az adatoknak.

A tárolt programokkal működő számítógép elvi felépítése funkcionálisan megfelel a ma használatos számítógépeknek:

- az adatok bevitelére és megjelenítésére önálló egységek szolgálnak;
- a gép tartalmaz egy vezérlőegységet, mely a tárolt program alapján irányítja a teljes gép működését;
- a gépnek része egy önálló műveletvégző egység, mely képes az összes fontosabb aritmetikai és matematikai logikai művelet végrehajtására;
- a számítógép rendelkezik egy belső tárolóval, ahol az aktuálisan szükséges programok és adatok tárolhatók.

A másik jelentős újítást a korábbi számológépek többségével szemben a bináris (kettes számrendszer alapján történő) adat (szám) és programutasítás tárolás és műveletvégzés jelentette.

A kettes számrendszer két számjegyéhez (0-hoz ill. 1-hez) ugyanis jól hozzárendelhetők két állapotú jelenségek (pl. van áram, vagy nincs áram egy ponton; mágnesezett vagy nem mágnesezett egy pont) az aritmetikai műveletek visszavezethetők a kétértékű (igaz, hamis) matematikai logika műveleteire, amelyek viszont logikai áramkörökkel modellezhetők.

Neumann János harmadik javaslata a számítógép soros utasításvégrehajtása volt. Ez azt jelenti, hogy a számítógép az utasításokat szigorúan egymás után, sorban hajtja végre (control flow). Ettől eltérés logikailag csak egy állapotjelző tartalmának vizsgálata alapján lehetséges, de ez is csak azt jelenti, hogy az utasítások végrehajtása a műveleti sor egy másik pontján folytatódik. „Természetesen a soros adatfeldolgozás lassúbb, mint a párhuzamos, de jóval kevesebb eszköz kell hozzá, és az eljárás lassúságát az igen gyors elektronika kiegyenlíti" – vélekedett Neumann János.

A soros utasítás-végrehajtásra példaként nézzük meg, milyen logikai műveletsorozattal tudjuk kiszámítani egy $a \cdot x^2 + b \cdot x + c = 0$ másodfokú egyenlet gyökeit. Az ismert $x_{12} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ képlet alapján a következő műveletsorozatot kell alkalmaznunk:

1. „*a*” beolvasása
2. Ha „*a*” értéke nem 0, folytasd az 5. lépéssel
3. Kiírás: „0-val nem lehet osztani”
4. Folytasd a 14. lépéssel
5. „*b*” beolvasása
6. „*c*” beolvasása
7. $d = b^2 - 4ac$ kiszámítása
8. Ha „*d*’ értéke nagyobb vagy egyenlő 0, folytasd a 11. lépéssel
9. Kiírás „Nincs valós gyök”
10. Folytasd 14. lépéssel
11. $x_1 = (-b + d) / 2 \cdot a$ kiszámítása
12. $x_1 = (-b - d) / 2 \cdot a$ kiszámítása
13. x_1, x_2 kiírása
14. A műveletsor befejezése

Végül térjünk vissza fejezetünk kiinduló kérdésére, hogy mivel magyarázható a számítógépek gyors elterjedése, miért alkalmazzák ezeket az eszközöket számtalan szakterületen?

- Ha egyszer a számítógép számára az ember megfogalmazott egy műveletsorozatot, akkor ezt a gép igen gyorsan képes végrehajtani, műveletvégző képessége az emberének sokszorosa. A tárolt programozás miatt az emberi adatközlés nem lassítja le a számítógép működését.
- Ha egy feladatot lefordítottunk a számítógép nyelvére, akkor az ennek megfelelő műveletsorozatot megtestesítő programot a számítógép bármeddig tárolni képes; ezt bármikor újra végre lehet hajtani eltérő kiinduló adatokkal is.

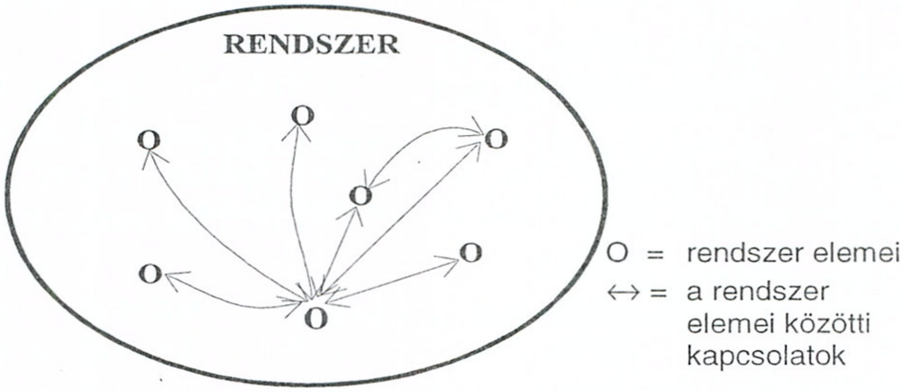
Így például, ha a másodfokú egyenletet megoldó előző programunkat tároljuk a számítógépen, akkor ezt felhasználva bármilyen másodfokú egyenlet megoldására alkalmassá tettük a gépet. (Ehhez csak a különböző „*a*”, „*b*” és „*c*” értékeket kell közölnünk a számítógéppel.) Arra is fontos felhívni a figyelmet, hogy ezt a másodfokú egyenletmegoldó programot bárkinek a számítógépére át lehet másolni, így a másodfokú egyenlet megoldását az összes számítógépet használó ember számára lehetővé tettük.

A számítógépeknek tehát az előzőekben részletezett tulajdonságai lehetővé teszik, hogy a megoldott feladatoknak megfelelő programokban megtestesülő szellemi értékek évről-évre felhalmozódjanak és így a számítógépek egyre több területen felhasználhatóvá válnak.

2. A rendszer fogalma

Rendszernek nevezzük a rendszer részeit alkotó elemek és ezek kapcsolatainak olyan együttesét, amelyek meghatározott ismérvek (vizsgálati célok) szempontjából összetartoznak.

A **rendszer elemei** alatt a rendszernek azokat az alkotóit értjük, melyeket a rendszer vizsgálata során már további részekre nem bontunk.



II. 2. sz. ábra
A rendszer szimbolikus ábrázolása

Látható, hogy definíciónk igen általános. Ilyen értelmezésben rendszernek tekinthetjük például a társadalmat, egy konkrét embert, egy gyárat vagy a UNIX operációs rendszert egyaránt.

A definícióval összefüggésben a következőkre hívjuk fel a figyelmet:

- Ha egy rendszert vizsgálunk, akkor mindig a valóság egy modelljét hozzuk létre. Ezért az, hogy mit tekintünk egy rendszernek, az a vizsgálat céljától függő elvonatkoztatás eredménye. Így például, ha egy számítógépes felhasználói programrendszert vizsgálunk, akkor ez a rendszer állhat a programegységekből és ezek funkcióiból, mint elemekből (például, ha a vizsgálat tárgya a rendszer felhasználói szolgáltatásai), de állhat programutasításokból is, mint elemekből (pl. ha hibakeresés a vizsgálat tárgya).
- Hangsúlyozni kell, hogy a rendszer nem egyszerűen elemeknek a halmaza, mindig beleértjük a rendszer fogalmába elemkapcsolatainak összességét is. Az elemkapcsolatok közül csak azokat tekintjük a rendszer részének, amelyek az adott vizsgálati cél szempontjából lényegesek. Így például, ha egy mun-

kaszervezet működési modelljét vizsgáljuk, mint rendszert, melynek elemei az adott szervezethez tartozó emberek, nem elemkapcsolat az a viszony, hogy a dolgozók közül ki született azonos városban.

Önszerveződő és adaptív egy rendszer, ha képes arra, hogy elemkapcsolatait külső beavatkozás nélkül megváltoztassa és ezáltal a környezetéhez alkalmazkodjon.

Önszabályzó egy rendszer, ha külső beavatkozás nélkül képes belső folyamatait irányítani.

Ilyen rendszerekre jó példa az emberi szervezet, egy gazdasági vállalkozás vagy egy automata gép. Az emberi szervezet adaptivitását mutatja például a meleg hatására való izzadás, az önszabályozást pedig a véráramlásunk folyamatos fenntartása.

A rendszer állapotába való külső beavatkozás típusa szerint megkülönböztetünk:

- **irányítást**, amikor a külső beavatkozás a rendszer állapotának változását céltudatosan befolyásolja.
- **szervezést**, amikor a külső beavatkozás a rendszer felépítését (struktúráját) változtatja meg.

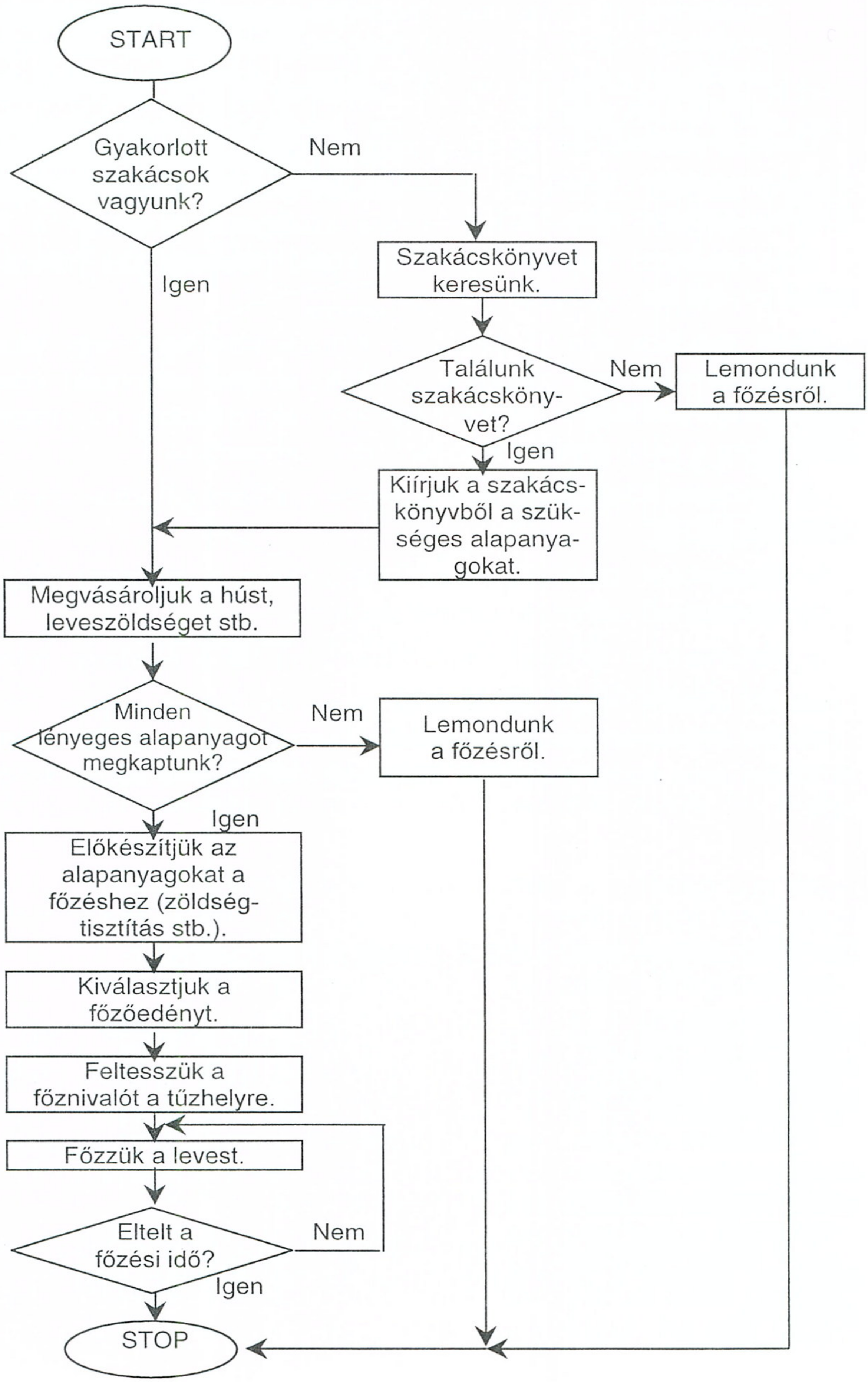
A külső beavatkozást közvetítő információt nevezzük **input**nak, a rendszer erre adott reakcióját megtestesítő információt pedig **output**nak.

3. Az algoritmus és a program fogalma

Ha egy feladatot kell megoldanunk, akkor ezt általában több részfeladatra bontjuk fel, melyekben különböző műveleteket végzünk el, feltételeket értékelünk ki, és így jutunk el célunkhoz.

Példaként nézzük meg, ha húslevest akarunk főzni, akkor milyen résztevékenységeken keresztül jutunk el az eredményhez és ennek során milyen döntéseket kell hoznunk.

Ezt egy úgynevezett folyamatábrán fogjuk szemléltetni, ahol téglalappal jelöljük a részműveleteket és fekvő rombusszal a döntési pontokat.


START
Gyakorlott szakácsok vagyunk?
Nem
Igen
Szakácskönyvet keresünk.
Találunk szakácskönyvet?
Nem
Lemondunk a főzésről.
Igen
Kiírjuk a szakácskönyvből a szükséges alapanyagokat.
Megvásároljuk a húst, leveszöldséget stb.
Minden lényeges alapanyagot megkaptunk?
Nem
Lemondunk a főzésről.
Igen
Előkészítjük az alapanyagokat a főzéshez (zöldségtisztítás stb.).
Kiválasztjuk a főzőedényt.
Feltesszük a főznivalót a tűzhelyre.
Főzzük a levest.
Eltelt a főzési idő?
Nem
Igen
STOP

A feladat elemzése alapján határozhatjuk meg azokat a lépéseket és döntési pontokat, melyeken keresztül végül elérhetjük a kívánt eredményt. Ezt a lépéssorozatot – beleértve a döntési helyzeteket is – szokták **algoritmusnak** nevezni. Ennek megfelelően tehát a pontos definíció a következő:

Algoritmusnak nevezzük egy feladat megoldását eredményező véges számú lépésben véget érő, egyértelmű szabályokkal megfogalmazható műveletsorozatot. Definíciónk értelmében tehát egy jól működő számítógépes program utasítássorozata és a π 5 tizedes pontosságú kiszámítására való matematikai eljárás is algoritmus.

A számítógépek lényegében mindig algoritmusokat végrehajtva működnek. Ebben az esetben azonban az alkalmazható elemi lépéseket a gép felépítése határozza meg.

Ha egy algoritmust a számítógép által értelmezhető és végrehajtható lépésekből építünk fel, akkor ezt **programnak** nevezzük, az elemi lépéseket pedig **utasításnak**. A program tehát, a számítógép számára értelmezhető és végrehajtható utasítások sorozata. Ezért a programkészítésnél olyan elemi lépésekre kell bontanunk a feladatot, melyek a számítógép nyelve által megengedett utasításokból állnak (lásd VI. fejezetben a számítógép utasításkészletét).

4. Információ és adat

Ha a környeztünkben zajló eseményeket befolyásolni akarjuk, azokra reagálni kívánunk, meg kell ismernünk azokat. Ezek az ismeretek hírek vagy közlemények formájában jutnak el hozzánk. A közleményeket mindig valamilyen hordozó közeg (fény, hang, szag stb.) által érzékeljük. Ha a közleménynek van olyan tartalma, amelynek révén olyan ismereteket szerzünk, amivel korábban nem rendelkeztünk, akkor a közlemény számunkra **információt** hordoz.

A közlemények meghatározott ideig rögzített formában is létezhetnek (például az újságban kinyomtatva, vagy a számítógép memóriájában tárolva), a hétköznapi életben ekkor beszélünk **adatokról**.

Hangsúlyozni kell, hogy az adat csak a közlemény formai oldalát jelenti, információt az ember számára csak értelmezésével jelenthet. Így például ugyanazon hangfrekvenciákat tartalmazó csengőhang az iskolában jelentheti az óra végét, a tűzoltóságnál pedig a tűzriadót.

Az elkövetkezendőkben az adat és az információ fogalmának pontosabb meghatározásával fogunk foglalkozni és bemutatjuk e két fogalom viszonyát.

Az IBM adatfeldolgozási szótára szerint:

Az adat tények, fogalmak, eligazítások olyan formai megjelenése (képe), amely alkalmas az emberi vagy az automatikus eszközök által történő kommunikációra, értelmezésre vagy feldolgozásra.

Az ISO szabványa szerint az információ az adatnak tulajdonított jelentés. Az információt úgy is szokták definiálni, hogy olyan tény, amely a befogadó ember számára új ismeretet tartalmaz, és ezáltal bizonytalanságunkat csökkenti.

Ezek szerint, pl. Kovács Béla APEH-nyilvántartásban tárolt vezeték- és utóneve adat, ugyanez Kovács Béla számára a személyi igazolványában szintén adat, míg az őt igazoltató rendőr számára (ha korábban nem ismerte Kovács Bélát) egyúttal információ is.

Az információ definíciójával összefüggésben lényeges hangsúlyozni, hogy az információ mindig kötődik az információt befogadó emberhez, az adat viszont személytelen, objektív ismeret. Ezt emeli ki a következő definíció: az adat értelmezhető (észlelhető, érzékelhető, felfogható és megérthető) ismeret, az információ pedig új ismeretként értelmezett adat.

Tehát például, ha nem tudunk kínaiul, akkor egy kínai könyvben szereplő mondatok számunkra nem jelentenek információt, mert nem vagyunk képesek értelmezni azokat. Ugyanakkor ez a könyv természetesen adatokat tartalmaz, mert kínaiul tudó emberek számára értelmezhető.

Az információ mérésére a matematikában (információelmélet) mérőszámot is kidolgoztak. Ennek megértéséhez vegyük példának azt az eseményt, hogy az utcán találkozunk egy emberrel és megkérdezzük az utónevét. Ha véletlenszerűen választunk, annak valószínűsége, hogy partnerünk a JÁNOS vagy az UBUL nevet fogja válaszolni, nyilván a nevek gyakoriságától függ. E nevek előfordulásának valószínűségét a következőképpen számíthatjuk ki:

$$p_J = \frac{\text{JÁNOS nevet viselők száma}}{\text{összes emberek száma}} \qquad p_U = \frac{\text{UBUL nevet viselők száma}}{\text{összes emberek száma}}$$

Ha valakiről csak azt tudjuk, hogy JÁNOS-nak hívják, nehezebben tudjuk megkeresni, beazonosítani az összes ember között, azaz ez kevesebb információt jelent számunkra. Ennek az az oka, hogy a

$p_J > p_U$. Azaz egy esemény egyedi információtartalmának nagysága (jelöljük I-vel) fordítottan arányos az adott esemény előfordulásának valószínűségével: $I_J < I_U$. Ezért logikusnak tűnik, hogy az

$$I_J \text{ arányos } \frac{1}{p_J} \text{ -vel} \qquad I_U \text{ arányos} \frac{1}{p_U} \text{ -vel}$$

szabályt állítsuk fel. E gondolatmenetet követve az információelmélet kidolgozójának, Shannon-nak a javaslata szerint egy esemény bekövetkezésének információtartalmát a

$$-\overset{2}{\log} p = \overset{2}{\log} \frac{1}{p}$$

mérőszámmal határozhatjuk meg. (Itt a 2-es alapú logaritmus csak a „skálabeosztást" befolyásolja.)

Végül ismételten érdemes arra felhívni a figyelmet, hogy az információ az értelmezőhöz, tehát az emberhez kötődő fogalom. Az adatokat feldolgozó eszköz, például a számítógép számára teljesen közömbös az adatok jelentéstartalma, vagyis az információ. Itt csak az a lényeges, hogy a számítógép a program formájában testet öltött algoritmust a szabályoknak megfelelően végrehajtsa. Tehát a számítógép mindig adatokkal és nem információkkal dolgozik.

5. Információtechnológia, információs és adatfeldolgozó rendszer, informatika

Az **információtechnológia** magába foglalja mindazon módszereket és eszközöket, melyek az információ előállítását, feldolgozását és továbbítását szolgálják. Ebben az értelemben információtechnológiai eszközöknek minősülnek:

- a hírközlő, kommunikációs és média eszközök,
- a számítástechnikai eszközök,
- az irodatechnikai eszközök (szervező eszközök, mikrofilm, másolók, iratmegsemmisítők stb.).

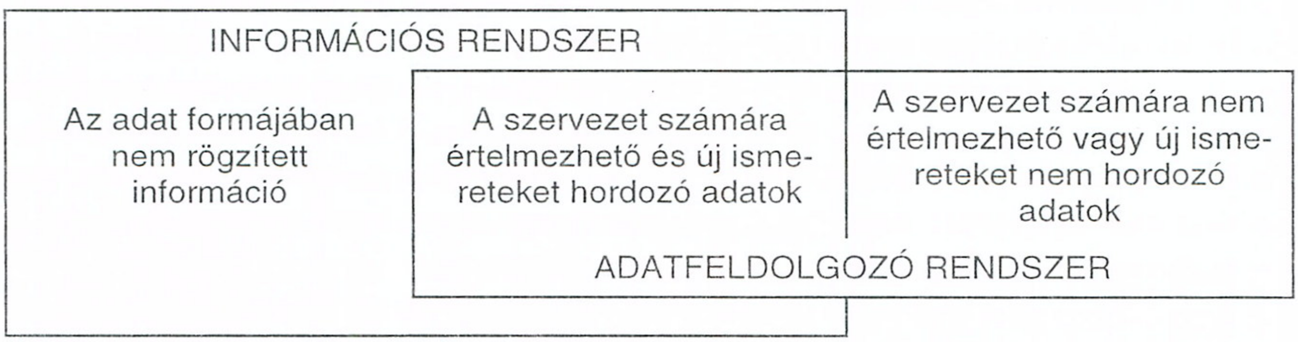
Az információtechnológia fogalmának megjelenése híven kifejezi azt, hogy a fenti három eszközcsoport vonatkozásában az utóbbi években határozott integrációs tendencia észlelhető.

A telefonközpontok ma már lényegében számítógépek, egy számítógépes rendszernek is szerves része a távadatfeldolgozó hálózat stb.

Az **információs rendszer** mindazon módszerek, eljárások, folyamatok és eszközök szervezett együttese, mellyel a szervezet tevékenységéhez információt állít elő, befogad, tárol, feldolgoz és továbbít.

Az **adatfeldolgozó rendszer** mindazon módszerek, eljárások és eszközök szervezett együttese, mellyel a szervezet adatot fogad, rögzít, feldolgoz, tárol, továbbít és megsemmisít.

Ha feltételezzük, hogy a szervezet céljaival és feladataival, valamint humán erőforrásaival teljes mértékben összhangban van a kialakított adatfeldolgozó rendszer (azaz az adatok mindig értelmezhetők és információt hordoznak), akkor az információs rendszernek mindig része az adatfeldolgozó rendszer. Ez azonban a valóságban legtöbbször nem így van, ezt szemlélteti a következő ábra:



II.3. sz. ábra
Az adatfeldolgozó és az információs rendszer viszonya

Vélhetően sokan tudnának gyakorlati példákat felsorolni olyan számítógépes rendszerekre, melyek esetenként senki által nem hasznosítható adatokat állítanak elő, vagy a szolgáltatott adatokat a kellően nem kiképzett felhasználók nem tudják értelmezni.

A definícióval kapcsolatban hangsúlyozni kell, hogy minden szervezetnek létezik információs és adatfeldolgozó rendszere, függetlenül attól, hogy a szervezet alkalmaz-e információtechnológiai eszközöket, vagy feladatait hagyományos nyilvántartások alapján végzi.

Most már eljutottunk az informatika fogalmának meghatározásához, bár meg kell jegyezni, hogy e téren még sok a terminológiai vita.

Informatikának nevezzük az információs rendszerek fejlesztésének, működtetésének, hasznosításának törvényszerűségével foglalkozó tudományágat (szakmát).

Sokan az informatikát a számítástechnikával azonosítják. Ez több szempontból is hiba:

- az információs rendszert leszűkítik az azt kiszolgáló eszközre, a számítógépes rendszerre,
- mint láttuk a számítástechnika az informatika eszközoldalának, az információtechnológiának csak egy részterülete.

Ez a helytelen szemlélet Magyarországon az elmúlt években igen komoly károkat is okozott. A rendszerfejlesztésben az eszközoldalra koncentrálnak és ennek során sokszor elfelejtették, hogy a szervezeti céloknak megfelelően olyan információrendszert kell kialakítani, melyet a szervezet képes befogadni. (A szervezet informatikai fogadóképessége a felhasználók képzettségétől kezdve az informatikusok szakértelmén keresztül, a vezetők szemléletéig számtalan tényezőtől függ.) Emiatt sok, például fejlettebb országokban bevált számítástechnikai eszköz hatékony alkalmazása is meghiúsult.

6. Kódolás, kódrendszer, kódkészlet

Az adatok megjelenési formája nagyon sokféle lehet, de a leggyakoribb az írott vagy nyomtatott szöveg; ennek egységelemét karakternek nevezzük.

A karakterek lehetnek

- kis és nagybetűk (A, B, C ... a, b, c ... y, z),
- számjegyek (0, 1, 2 ... 9),
- különleges jelek (+, -, ? ... stb.).

A csak számjegyekből álló karaktersorozatokat **numerikus**nak, a csak betűkből állókat **alfabetikus**nak, a fenti három típust vegyesen tartalmazó karaktersorozatokat pedig **alfanumerikus**nak nevezzük. Például a „1233” karakterből álló adat numerikus, a „1+3=C6” karakterekből álló pedig alfanumerikus adat.

Az adatokat a különböző adathordozókon olyan formában célszerű rögzíteni, hogy helyigényük ne legyen túlzott mértékű, és minél kevesebb hibát kövessünk el rögzítéskor. Ezt többek között az adatok kódolásával érhetjük el.

Ezen túlmenően például az adatok számítógépes feldolgozásához az ember által megszokott formátumú tízes számrendszerű számokat és karaktersorozatokat is át kell alakítani a számítógép működésének megfelelő 0 és 1-ből álló jelsorozattá.

Az egyes hírekben és közleményekben lévő adatok formai átalakítását **kódolás**nak nevezzük. Ennél általában követelmény, hogy az adatok kódolás előtti és utáni formája között kölcsönösen egyértelmű megfeleltetés legyen.

A kódolás tehát áttérést jelent egy jelkészlet és ezzel összefüggő szabályrendszer használatáról egy másik jelkészletre és szabályrendszerre.

Erre mutat példát a következő ábra

			0	⟷	0 0 0 0
fekete	⟷	0	1	⟷	0 0 0 1
kék	⟷	1	2	⟷	0 0 1 0
zöld	⟷	2	3	⟷	0 0 1 1
világoskék	⟷	3	4	⟷	0 1 0 0
piros	⟷	4	5	⟷	0 1 0 1
lila	⟷	5	6	⟷	0 1 1 0
sárga	⟷	6	7	⟷	0 1 1 1
fehér	⟷	7	8	⟷	1 0 0 0
			9	⟷	1 0 0 1
	a)			b)	

II.4. sz. ábra

a) Színek kódolása; b) A tizedes számrendszer számjegyeinek bináris (kettes számrendszerű) kódolása

Az ábra b) részében bemutatott bináris kódolás esetén, ha a közlemény a 1910 karaktersorozat, akkor ennek kódolás utáni megfelelője

0 0 0 1	1 0 0 1	0 0 0 1	0 0 0 0
1	9	1	0

A kódolás során használt jelkészletet és formai szabályrendszert együttesen **kódrendszer**nek nevezzük.

Az előző ábra a) részének példájában az alkalmazott jelkészlet (1, 2, 3, 4, 5, 6, 7), az áttérés formai szabálya pedig az ábráról olvasható le (fehér ⟷ 1, kék ⟷ 2 stb.).

7. A hardver, szoftver és a förmver fogalma

A hardver eredeti angol jelentése „kemény áru” (vas-áru). Lényegében ez a számítógépet alkotó, kézzel fogható eszközök összefoglaló neve, a számítógép elektronikus áramköreit, mechanikus berendezéseit, kábeleit, csatlakozásait és perifériáit nevezzük így.

A hardver önmagában egy működésképtelen eszközhalmaz. Ahhoz, hogy a számítógép egy feladatot meg tudjon oldani, azt algoritmus formájában meg kell fogalmazni és a számítógéppel az általa értelmezhető utasítások formájában közölni kell. A számítógépet működőképessé tevő programok összességét szoftvernek (eredeti jelentése: „lágy áru") nevezzük.

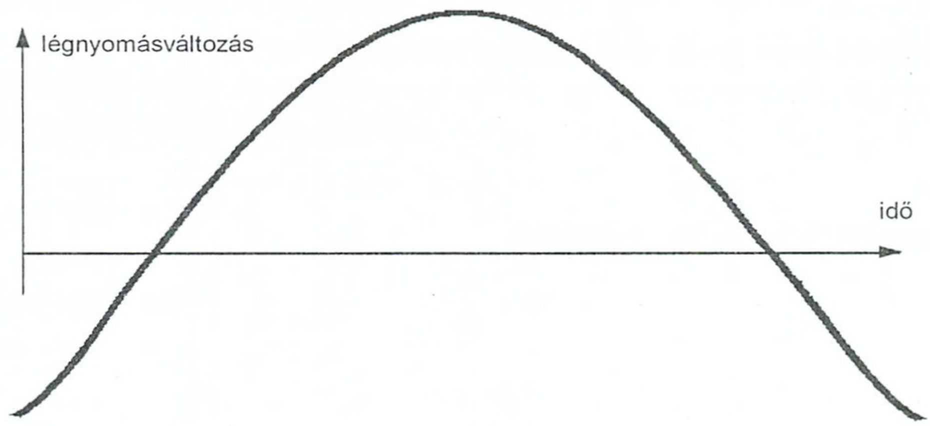
A számítógépek általános, gyakran ismétlődő vezérlési feladatait végző programokat legtöbbször kisebb, csak olvasható memóriákban (ROM – Read Only Memory) helyezik el. Ezeket a programokat, a tárolóeszközzel együtt förmvernek (firmware) nevezik.

B. AZ ANALÓG ÉS DIGITÁLIS TECHNIKA

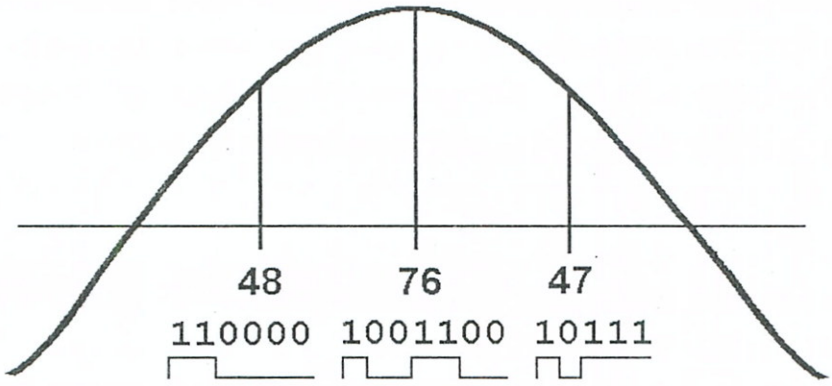
A digitális technikával abból a világból, amiben a megszokott mércerendszerünk „analóg" a valóságosan érzékelhető mennyiségekkel, átlépünk a számoknak egy olyan világába, ahol a folyamatosan változó mennyiségeket diszkrét módon, pl. számjegyekkel ábrázolva dolgozzuk fel.

Talán a legjobb példa erre az emberi hang.

A II.5. sz. ábrán látható az emberi hang ábrázolása, ahogy az a valóságot megközelíti, a II.6. sz. ábrán pedig számjegyekkel meghatározott értékekkel megjelenítve ugyanaz a hang.



II.5. sz. ábra: Emberi hangot ábrázoló görbe



II.6. ábra: Emberi hangot ábrázoló görbéből vett minták alapján előállított digitális (azaz számjegyekkel meghatározott értékekkel megjelenített) jelek

Az ábrák alapján is egyszerűen megítélhető, hogy mennyivel nagyobb nehézségi fokot jelent az emberi hang alakhű ábrázolása annál, mintha csak azt kell előállítanunk valamilyen elektronikus eszközzel, hogy van jel vagy nincs jel. Ezzel érzékelhetjük azt, hogy mi a különbség az analóg és digitális jel között és hogyan alakíthatjuk át az egyik jelet a másikba.

1. Analóg technika

Analógnak nevezzük az olyan eszközöket, eljárásokat stb., amelyek folytonos mennyiségeket ábrázolnak, illetve dolgoznak fel. (A szó alapjelentése: hasonlóságon alapuló.) Az analóg jelek ábrázolási vagy feldolgozási tartománya megfeleltethető a valós számok egy intervallumának.

Az analóg jelenségeket ezért folytonos fizikai mennyiségek segítségével modellezhetjük. (Egy analóg jel bizonyos határok között a jeltartományon belül tetszőleges értéket felvehet.)

Analóg elven működő áramkörök:

- lineáris erősítők: műveleti erősítők, hangfrekvenciás erősítők, video erősítők, széles sávú erősítők stb.,
- nem lineáris erősítők: keverők, modulátorok, függvényeket megvalósító áramkörök, komparátorok stb.,
- forrásáramkörök: oszcillátorok, stabilizátorok stb.,
- jelátalakító áramkörök: analóg-analóg jelátalakító, analóg-digitális jelátalakító, digitális-analóg jelátalakító stb.

2. Digitális technika

Digitálisnak nevezzük az olyan adatokat, eszközöket, eljárásokat stb., amelyek változó mennyiségeket számjegyekkel, diszkrét módon ábrázolnak, illetve dolgoznak fel.

Az **analóg-digitális átalakítók** (A/D átalakító, A/D konverter, ADC, digitalizáló) olyan áramkörök, illetve eszközök, amelyek a bemenetükre adott analóg jelet digitális jellé alakítják át.

A **digitális ábrázolás** valamely változó értékének diszkrét ábrázolása számjegyekkel. (A digitális jel számjegyekkel azért ábrázolható, mivel egy ilyen jel csak véges számú diszkrét értéket vehet fel.)

A digitális elven működő áramkörök a digitális ábrázolású adatok feldolgozására szolgálnak. A digitális áramkörök impulzusok alakjában lévő adatokat dolgoznak fel. A pozitív és negatív, azaz bináris (binary) impulzusokat feldolgozó áramkörök kombinációs vagy szekvenciális működésűek lehetnek.

3. Jelátalakítás

3.1. Analóg–digitális átalakítás

Az analóg–digitális (A/D) átalakítás olyan eljárás, amely az analóg jelből előállítja az annak megfelelő digitális jelet. Általában ez két műveletből áll: a mintavételből és a kvantálásból.

Nézzük, hogy esetünkben mi is az a minta?

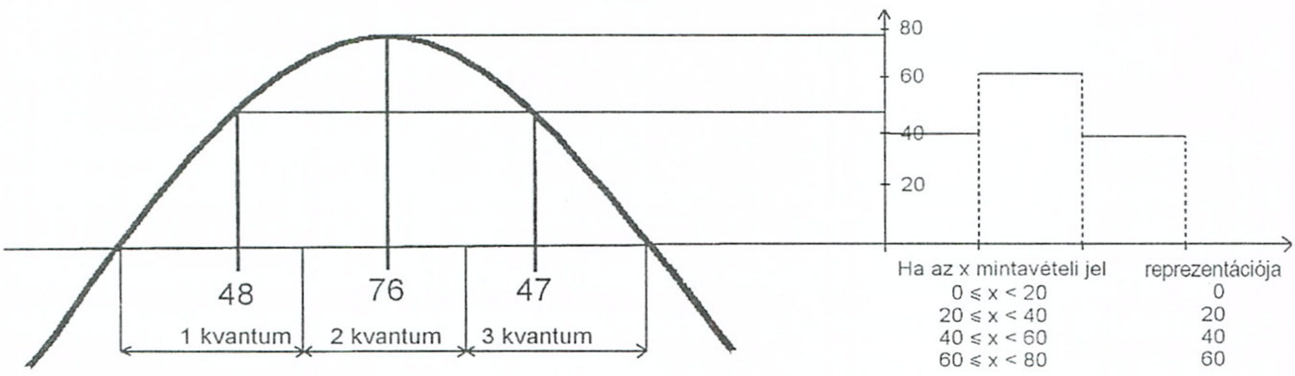
Mintának nevezzük valamely jel valamely jellemzőjének megmért értékét egy adott pillanatban.

A mintavétel az az eljárás, amellyel egy folytonosan változó menynyiséget csak bizonyos időpontokban (a mintavételi időpontokban) felvett értékeivel jellemzünk.

Ez az alapja az analóg jelek digitális átalakításának, amikor is az analóg jelet (pl. beszéd) kellő mintavételi frekvenciával letapogatják. (A mintavételi frekvencia, vagyis a mintavétel gyakorisága, a másodpercenként vett minták számát jelenti.)

A mintavétel után a digitalizálás következő művelete a kvantálás.

A kvantálás során a változó értéktartományát egymásba nem nyúló, nem feltétlenül egyenlő hosszúságú véges sok intervallumra osztják, és minden intervallumot egy kijelölt elemével reprezentálnak. (Ettől lesz a jel szakaszos.)



II.7. sz. ábra: A mintavétel során kapott értékek digitalizálása kvantálással

3.2. Digitális–analóg átalakítás

A digitális–analóg (D/A) átalakítás olyan eljárás, amely a digitális jelből előállítja az analóg jelet.

A digitális/analóg átalakító (D/A átalakító, D/A konverter, DAC) a digitális ábrázolású adatot analóg ábrázolású adattá alakítja, vagyis olyan áramkör, amelynek bemenetére (bemeneteire) digitális jelet (jeleket) adnak és kimenetén a bemeneti jel számértékének megfelelő nagyságú analóg jel jelenik meg. Erre jó példa a számítógépen digitális formában tárolt adatok megjelenítése analóg elektronsugárral a monitor képernyőjén.

4. Logikai áramkörök

4.1. Bináris logika

A **logikai áramkörök** diszkrét értékű bemenő jelekből, meghatározott matematikai logikai függvényeknek megfelelő kimenőjelet állítanak elő. Ezek logikai értékeket az áramkör valamely elektromos jellemzőjéhez rendeljük hozzá.

Egy logikai áramkör kapcsolási funkciót lát el, vagyis megengedi vagy megakadályozza az áram folyását. A számítógépen belül ez a funkció automatikusan végbemegy, amikor például az órajel megje-

lenik. Kapcsolóáramkörök kombinációja logikai műveletek végrehajtására alkalmas.

Általában kétértékű logikai rendszereket használnak, ahol leggyakrabban a feszültség a választott elektromos jellemző. Ezek a kapcsolóéhoz hasonló jellegű működéssel a logikai döntéseket elektromosan valósítják meg. Ez a bináris működés minimális követelményeket támaszt az alkatrészek tűrésével szemben, ezért a digitális áramkörök integrált formában jól megvalósíthatók.

Az olyan logikai rendszert, amely **kétállapotú** elemekkel valósítja meg a kívánt logikai kapcsolatokat, **bináris logikának** nevezzük.

A két különböző állapot szokásos elnevezései:

H (High=Magas)	L (Low=Alacsony)
„1"	„0"
„IGAZ"	„HAMIS"
„IGEN"	„NEM"
„TRUE"	„FALSE"

A kétértékű logikai rendszerekben **két logikai szint** lehetséges: a feszültség abszolút értelemben nagyobb értéke a logikai magas szint, jele H (H=High), kisebb értéke a logikai alacsony szint, jele L (L=Low), a kettő közötti átmeneti tartomány biztosítja a jelek jó szétválaszthatóságát.

A pozitív logikai konvenció a fizikai változókhoz (áram, feszültség) a H=1 és L=0 szerint, a negatív logikai konvenció a H=0 és az L=1 szerint rendeli hozzá a logikai értékeket.

A logikai áramkör bemenetei és kimenetei a két szint közötti értéket hibátlan működés esetén csak átmeneti (tranziens) állapotban vehetnek fel.

A logikai áramkörök a **három alapvető logikai funkciót** , az **„ÉS" (AND)**, a **„VAGY" (OR)** és a **„NEM" (NOT)** műveleteket végzik el. Ezzel a három alapművelettel összetettebb logikai kifejezések írhatók fel.

4.2. Boole-algebra

George Boole (1815–1864) angol matematikus alakította ki azt az algebrai rendszert, amely logikai feladatok megoldására is alkalmas. Ezzel a rendszerrel megteremtették a mai digitális elektronikus berendezések kialakításának elméleti alapjait.

A Boole-algebrában egy nem üres M halmazból indulunk ki, és feltesszük a következőket:

M-en értelmezve van két kétváltozós művelet, az egyiket nevezzük összeadásnak (+), a másikat szorzásnak (·). Ez azt jelenti, hogy M bármely A és B elemére $A + B$ is, $A \cdot B$ is eleme M-nek.

M-en értelmezve van egy egyváltozós művelet, nevezzük komplemensképzésnek ($\overline{\ }$), vagyis M bármely A eleme esetén $\overline{A}$ is eleme M-nek.

Az M halmazt a fenti műveletekkel együtt Boole-algebrának nevezzük, ha M bármely A, B, C elemére fennállnak a következők:

(1) $A \cdot (B \cdot C) = (A \cdot B) \cdot C$;

(2) $A + (B + C) = (A + B) + C$;

(3) $A \cdot B = B \cdot A$;

(4) $A + B = B + A$;

(5) $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$;

(6) $A + (B \cdot C) = (A + B) \cdot (A + C)$;

(7) $A \cdot (A + B) = A$;

(8) $A + (A \cdot C) = A$.

(9) Létezik M-nek olyan, a Φ-val jelölt eleme, hogy minden M-beli A-ra
$\Phi \cdot A = \Phi$;
$\Phi + A = A$.

(10) Létezik M-nek olyan, a Φ-tól különböző, I-vel jelölt eleme, hogy minden M-beli A-ra
$I \cdot A = A$;
$I + A = I$

(11) Bármely M-beli A esetén
$A \cdot \overline{A} = \Phi$;
$A + \overline{A} = I$.

A fenti (különben nem független) axiómáknak több konkrét matematikai struktúra is megfelel.

A Boole-algebráknak alapvető szerepük van a halmazelméletben, a valószínűségszámításban, továbbá a mi tárgyalásunkat közelebbről érintő kétértékű logikai változókkal kapcsolatos műveleteknél is.

A **logikai algebra** számunkra legfontosabb alkalmazása a G. E. Shannon által kidolgozott **kapcsolás-algebra**, amely a **digitális áramkörök működésének alapját képezi**. Itt az M halmaznak két eleme van, a 0 és az 1.

4.2.1. Boole-művelet

A Boole-művelet olyan logikai művelet, amely logikai értékekhez (a művelet operandusaihoz) logikai értékeket (a művelet eredményét) rendel hozzá. Pl.: a kétoperandusú ÉS művelet az A és B logikai változókhoz az alábbi eredményt (A · B) rendeli:

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

A legfontosabb Boole-műveletek az egyoperandusú (egyváltozós) NEM művelet (negáció) és a kétoperandusú (kétváltozós) ÉS, VAGY, illetve KIZÁRÓ VAGY műveletek.

4.2.2. Igazságtáblázat

A Boole-műveleteket a logikában általában ún. **igazságtáblázatukkal** adják meg. Ez egy olyan táblázat, amely az operandus(ok), más néven változók minden lehetséges értékkombinációjához megadja az eredmény értékét.

A lehetséges egy- és kétoperandusú Boole-műveleteket az alábbi igazságtáblák foglalják össze:

Egyváltozós műveletek igazságtáblázata

Bemenet értéke	Kimenet értéke			
	0 állandó	EGYEN-ÉRTÉKŰ-SÉG	NEM	1 állandó
0	0	0	1	1
1	0	1	0	1

Kétváltozós műveletek igazságtáblázata (két bemenő változó esetén a lehetséges logikai függvények):

A bemenet értékei		Konstans 0	NEM-VAGY	A tiltása ÉS	Egyváltozós NEM (A)	B tiltása ÉS	Egyváltozós NEM (B)	Kizáró VAGY (ANTIVALENCIA)	NEM-ÉS	ÉS	Egyenértékűség (EKVIVALENCIA)	Egyváltozós egyenértékűség (B)	Ha A akkor B (IMPLIKÁCIÓ)	Egyváltozós egyenértékűség (A)	H B akkor A (IMPLIKÁCIÓ)	Megengedő VAGY	Konstans 1
A	B	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

II.2. sz. táblázat
Egy- és kétváltozós Boole-műveletek igazságtáblázatai

A logikai függvények táblázatai tehát megadják, hogy a bemeneti jelek különféle variációi a kimeneten milyen jelet eredményeznek.

Bebizonyítható, hogy minden logikai művelet kifejezhető az ÉS, a VAGY és a NEM műveletek alkalmas kombinációjával.

Egyes lehetséges kimeneteknek az áramköröknél fontos KIZÁRÓ VAGY (jelölése $\oplus$) az ÉS (jelölése $\cdot$), VAGY (jelölése +) és NEM (jelölése ¯) műveletekkel az alábbi függvények feleltethetők meg:

kimenet sorszáma	függvény	kimenet sorszáma	függvény
0	0	8	$A \cdot B$
1	$\overline{A + B}$	9	$\overline{A \oplus B}$
2	$\overline{A} \cdot B$	10	B
3	$\overline{A}$	11	$\overline{A} + B$
4	$A \cdot \overline{B}$	12	A
5	$\overline{B}$	13	$A + \overline{B}$
6	$A \oplus B$	14	$A + B$
7	$\overline{A \cdot B}$	15	1

II.3. sz. táblázat
Egy- és kétváltozós logikai függvények

Az igazságtáblázatokkal előbb megadott műveletek eleget tesznek a Boole-algebra szabályainak.

4.3. Kapuáramkörök, logikai áramkörök

A **kapuáramkörök** olyan egy vagy több-bemenetű logikai áramkörök, amelyek egy meghatározott logikai műveletet valósítanak meg. A logikai felépítéstől függően a kapuk lehetnek kombinációs, vagy szekvenciális (sorrendi) áramkörök.

A **kombinációs** áramkör olyan logikai áramkör, amely nem tartalmaz tárolóelemet, és amelynek kimeneti értékei csakis a bemeneti értékek pillanatnyi értékétől függenek.

A **szekvenciális (sorrendi)** áramkörök következő állapotát a jelen bemeneti állapot és a hálózat belső állapota együttesen határozza meg.

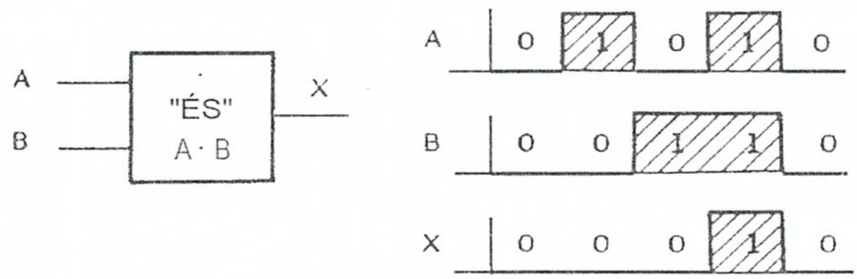
ÉS (AND) kapu

Az **ÉS kapu** olyan logikai áramkör, amely az **ÉS** műveletet valósítja meg, vagyis az adott bemeneti logikai változókra az **ÉS** művelet eredményét képezi.

A logikai szorzást végző **ÉS** kapuáramkör olyan több-bemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg az 1-et (logikában az igazat) fizikailag képviselő jelszint (továbbiakban **H** szint), ha az összes bemenetén **H** szint van.

A logikai szorzás (konjunkció) igazságtáblázata (lásd logikai függvénytáblázat, 8-as számú függvény):

A	B	$X = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1



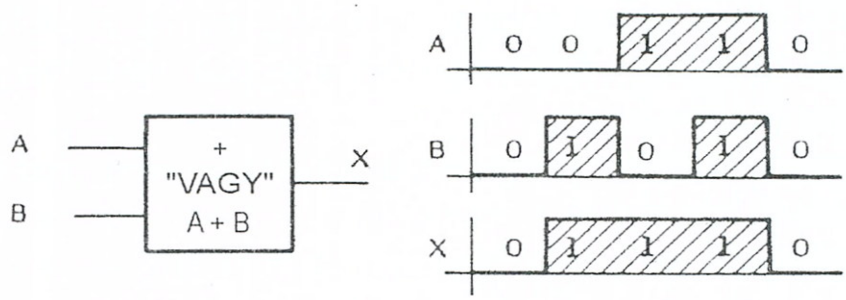
II.8. sz. ábra: A logikai szorzás jelképi jelölése és idődiagramja

VAGY (OR) kapu

! A logikai összeadást végző **VAGY** kapuáramkör olyan többbemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg **H** szint, ha legalább egy bemenetén **H** szint van.

A logikai összeadás (diszjunkció) igazságtáblázata (lásd logikai függvénytáblázat, 14-es számú függvény):

A	B	X = A + B
0	0	0
0	1	1
1	0	1
1	1	1



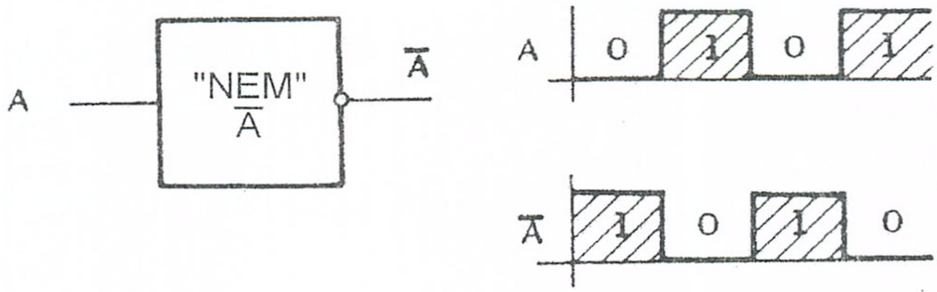
II.9. sz. ábra: A logikai összeadás jelképi jelölése és idődiagramja

INVERTER vagy NEM (NOT) kapu

! A **NEM** kapu olyan logikai áramkör, amely a **NEM** műveletet valósítja meg, azaz bemenetére adott logikai változóra a **NEM** műveletének eredményét képezi.

A **NEM** kapuáramkör egybemenetű logikai elem, amelynek a kimenetén mindig a bemenettel ellentétes szint jelenik meg (a **H** szinttel ellentétes szint jele: **L**).

Ha $A = 1$ akkor $\overline{A} = 0$ (L szint),
ha $A = 0$ akkor $\overline{A} = 1$ (H szint).



II.10. sz. ábra: Az invertálás jelképi jelölése és idődiagramja

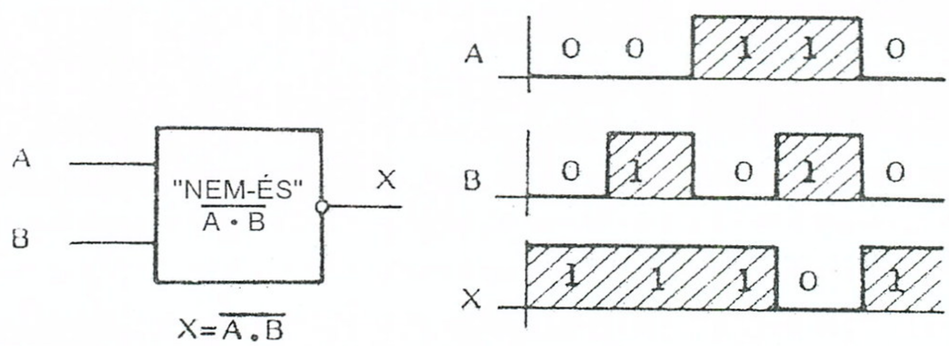
NAND vagy NEM-ÉS (NOT-AND) kapu

A **NAND** kapu olyan logikai áramkör, amely a **NEM-ÉS** műveletet valósítja meg, vagyis az adott bemeneti logikai változókra a **NEM-ÉS** művelet eredményét képezi.

A **NAND** kapuáramkör több-bemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg **L** szint, ha az összes bemenetén **H** szint van.

A **NEM-ÉS** művelet igazságtáblázata (lásd logikai függvénytáblázat, 7-es számú függvény):

A	B	$X = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



II.11. sz. ábra. A NEM-ÉS művelet jelképi jelölése és idődiagramja

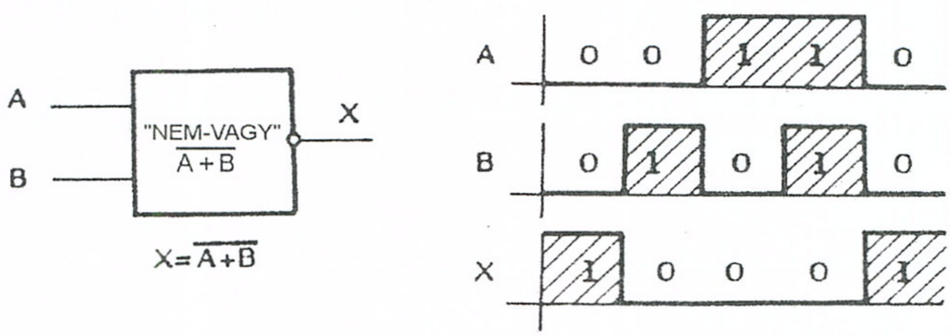
NOR vagy NEM-VAGY (NOT-OR) kapu

! A NOR kapu olyan logikai áramkör, amely a NEM-VAGY műveletet valósítja meg, vagyis az adott bemeneti logikai változókra a NEM-VAGY művelet eredményét képezi.

A NOR kapuáramkör több-bemenetű logikai elem, amelynek a kimenetén akkor és csak akkor jelenik meg L szint, ha legalább egy bemenetén H szint van.

A NEM-VAGY művelet igazságtáblázata (lásd logikai függvénytáblázat, 1-es számú függvény):

A	B	$X = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



II.12. sz. ábra. A NEM-VAGY művelet jelképi jelölése és idődiagramja

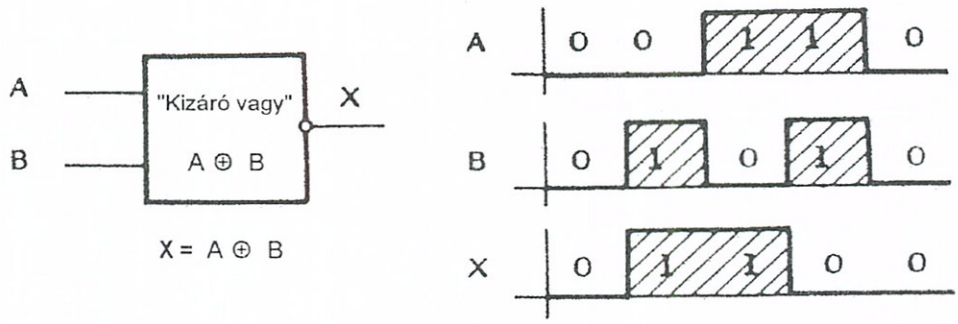
A KIZÁRÓ VAGY (EXCLUSIVE OR, XOR)

! **A kizáró VAGY** olyan logikai áramkör, mely az ún. **ANTIVALENCIA** műveletet valósítja meg; a bemenetére adott logikai változók az **ANTIVALENCIA** műveletének eredményét képezik.

A kizáró VAGY kétbemenetű logikai elem, amelynek kimenetén akkor és csak akkor jelenik meg **H** szint, ha csak egy bemenetén van **H** szint.

A kizáró VAGY igazságtáblázata (lásd logikai függvénytáblázat, 6-os számú függvény):

A	B	$X=A\oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



II.13. sz. ábra. A logikai kizáró VAGY jelképi jelölése és idődiagramja

Kombinációs logikai hálózatok

A kombinációs döntési hálózat általában olyan digitális hálózat, melynek bármely bemeneti állapotához, jelkombinációjához egy definiált kimeneti jelkombináció tartozik.

A mikroszámítógépek általában nem rendelkeznek az összes alapkapuáramkörrel, de **kombinációs hálózatokkal minden szükséges logikai függvény előállítható.**

Az

$$\overline{A\cdot B} = \overline{A}+\overline{B},$$
$$\overline{A+B} = \overline{A}\cdot\overline{B}$$

ún. de Morgan-féle szabályok segítségével például az **ÉS** művelet előállítható a **VAGY** és **NEM** művelet segítségével:

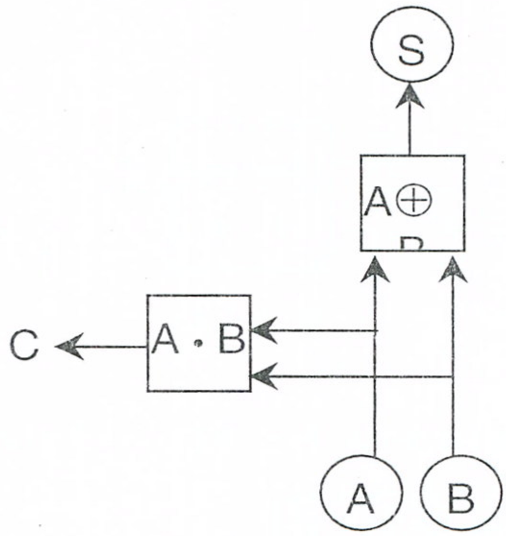
$$A\cdot B = \overline{\overline{A}+\overline{B}}.$$

A számítógépek működésében meghatározó jelentőségű az aritmetikai műveletek végrehajtása. Ezért nagyon fontos, hogy aritmetikai műveleteket végrehajtó áramköröket is kialakíthatunk kombinációs áramkörökből.

Erre példát a következő 1-bites összeadó egység áramköri felépítése mutat.

Az 1-bites összeadás igazságtáblázata

A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1



Jelölések:
A, B a két összeadandó érték
S az összeg
C a következő helyi értékre való átvitel

C. DIGITÁLIS ADATÁBRÁZOLÁS A SZÁMÍTÓGÉPEKBEN

1. Számrendszerek

A számrendszerek valós ábrázolására szolgáló jelek és az ezek alkalmazására vonatkozó szabályok összessége.

Legelterjedtebben a helyi értékes rendszereket használják, amelyekben a valós számot karakterek sorozata úgy ábrázolja, hogy minden számjegypozícióhoz valamely helyi érték van hozzárendelve, és a valós szám értékét az egyes helyi értékek és a hozzájuk tartozó alaki értékek szorzatainak az összege adja.

A digitális technikában a számrendszereknek alapvetően fontos gyakorlati szerepük van. A mennyiségeket (valós számokat) a helyi értékes írásmódban a választott alapszám hatványaival írjuk fel. Az alapszám bármely 1-nél nagyobb egész szám lehet.

1.1. A tízes (decimális) számrendszer

A tízes számrendszer alapszáma a 10.

A tízes számrendszerben a valós számot karakterek sorozata ábrázolja úgy, hogy az egyes számjegypozíciókhoz a tizedesjeltől balra a 10^0, 10^1, 10^2 stb. helyi értékek, míg a tizedesjeltől jobbra a 10^{-1}, 10^{-2}, 10^{-3} stb. helyi értékek vannak hozzárendelve, és a valós szám értékét az egyes számjegypozíciókon álló számjegyek és a hozzájuk tartozó helyi értékek szorzatainak összege adja.

Pl.: a 135, illetve a 123,45 karaktersorozat értéke a tízes számrendszerben:

$$135 = 1 \times 10^2 + 3 \times 10^1 + 5 \times 10^0$$
$$123{,}45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

1.2. A kettes (bináris) számrendszer

A kettes számrendszer alapszáma a 2. A kettes számrendszerben egy számjegyet két számjegy, a „0" és az „1" segítségével lehet felírni, a helyi értékkel tüntetve fel azt, hogy a jegyet a kettő hányadik egész kitevőjű hatványával kell megszorozni. Például:

Decimális írásmód	Írásmód a 2 alapú hatványok összegével	Bináris írásmód
1	1×2^0	1
2	$1 \times 2^1 + 0 \times 2^0$	10
3	$1 \times 2^1 + 1 \times 2^0$	11
4	$1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	100
5	$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	101
6	$1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	110
7	$1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	111
8	$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	1000
135	$2^7 + 2^2 + 2^1 + 2^0$	10000111
0,25	2^{-2}	0,01

Mint említettük, az elektronikus digitális számítógépek a kettes számrendszer alapján működnek, amit az a tény indokol, hogy bizonyos áramköri elemek és áramkörök tulajdonságai alapján két állapot jól meghatározható, pl.: mechanikus vagy elektronikus kapcsoló (félvezető eszközök, flip-flop áramkör, kriotron, ferrit stb.) eszközökkel.

A kettes számrendszerben megadott számokat vagy sorosan impulzussorozatokkal, vagy párhuzamosan, az egyes helyi értékeknek megfelelő impulzusokkal lehet reprezentálni.

1.3. A tizenhatos (hexadecimális) számrendszer

A tizenhatos (hexadecimális) szám olyan helyi értékes számrendszerben ábrázolt szám, amelynek alapszáma 16.

A hexadecimális karakterek szokásos jelölése a következő:

- decimálisan: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
- hexadecimálisan: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Tehát a 10 decimális számjegy mellett még az ABC első hat betűjét is felhasználják a 16 számjegy jelölésére.

Egy hexadecimális számjegy négy bináris számjeggyel ábrázolható: $(16=2^4)$.

Példa: Legyen egy decimális szám: 26, ennek bináris ábrázolása: 11010, hexadecimális számmal ábrázolva: 1A. A számítógépek tárolóegységét, a bájt 8 bitjét, két hexadecimális számjeggyel lehet leírni.

Például: a 1001 1111 bájt értéke hexadecimálisan kifejezve: 9F.

Példák a számok különböző számrendszerekben való ábrázolására:

Decimális 10-es	Bináris 2-es	Hexadecimális 16-os
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

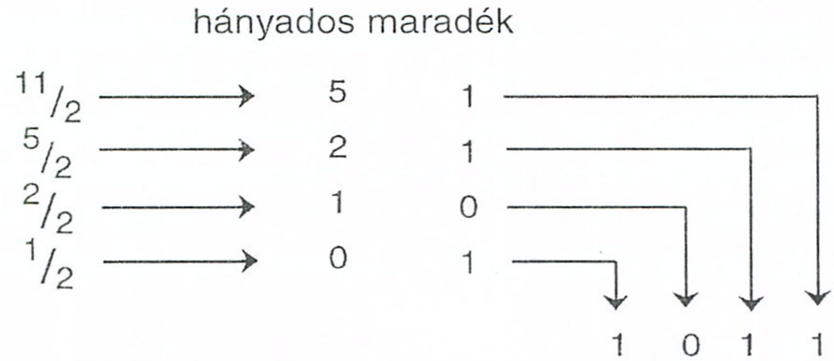
II.4. sz. táblázat
Példák számok 10-es, 2-es és 16-os számrendszerben történő ábrázolására

1.4. Számrendszerek átalakítása

1.4.1. Decimális–bináris átalakítás

A decimális–bináris átalakítás az az eljárás, amelynek során a tízes számrendszerben írt számokat azokkal egyenlő értékű kettes számrendszerbeli számokká alakítják át.

Az átalakítás algoritmusát a II.14. sz. ábra szemlélteti.



$11_{10} = 1011_2$

II.14. sz. ábra: Számok átalakítása
10-es számrendszerből 2-es számrendszerbe

1.4.2. Bináris–decimális átalakítás

A bináris–decimális átalakítás az az eljárás, amelynek során a kettes számrendszerben írt számokat azokkal egyenlő értékű tízes számrendszerbeli számokká alakítják át.

Például a 1011 bináris szám esetén az eljárás a következő:

$$1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 =$$
$$= 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 = 11_{10}$$

1.5. Műveletvégzés bináris számokkal

1.5.1. Bináris számok összeadása

A lehetséges kombinációk két bináris szám összeadása esetén:

1. Példa:

a + b	=	összeg	átvitel
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	10	1

Az átvitel a következő magasabb helyi értékű pozícióhoz kerül hozzáadásra.

2. Példa: a decimális és bináris számok összeadásának összehasonlítása:

	decimális összeadás	bináris összeadás
	208	11010000
	+ 92	+ 1011100
összeg:	300	100101100
átvitel:	11	11 1

1.5.2. Bináris számok kivonása

Két szám különbségét a számítógép ún. komplemensképzési és összeadási művelettel állítja elő.

A számítógépben a bináris számok ábrázolására korlátozott számú számjegypozíció, másnéven bit áll rendelkezésre. Legyen az adott számú pozíción ábrázolható legnagyobb szám M. Ekkor egy tetszőleges bináris szám **egyes komplemensének** az M és az adott szám különbségét értjük. **Kettes komplemensnek** nevezzük az előző különbségnél eggyel nagyobb számot.

A meghatározásból következik, hogy ha egy számhoz hozzáadjuk a kettes komplemensét, éppen azt a legkisebb számot kapjuk, amely az adott számú pozíción már nem fér el.

Például: Tegyük fel, hogy a számok ábrázolásához 4 bit áll rendelkezésre. Határozzuk meg a 11 bináris kódú szám egyes és kettes komplemensét. A négy biten ábrázolható legnagyobb szám: 1111.

	1111		1100
	- 0011		+ 1
Egyes komplemens:	1100	Kettes komplemens:	1101

Ellenőrizzük, hogy a kettő összege valóban az a legkisebb szám, ami nem fér el 4 biten.

A legkisebb szám, ami már 4 biten nem fér el: 10000.

Az eredeti szám:	0011
Kettes komplemens:	+ 1101
	10000

A kettes komplemens tehát a számot éppen arra a legkisebb számra egészíti ki, amely az adott számú pozíción már nem fér el. A

fenti összeadást érdemes még egyszer szemügyre venni. Ha a műveletet a gép végzi, az eredmény 0000 lesz, hiszen csak 4 bit áll rendelkezésre, és a gép a bal oldali egyest egyszerűen levágja. Ha tehát egy számhoz hozzáadjuk a kettes komplemensét, eredményül 0-át kapunk (feltéve, hogy korlátozott számú pozíció áll rendelkezésre). A kettes komplemenst tehát tekinthetjük az eredeti szám belső ábrázolású ellentettjeként. A gondot most már csak az jelenti, hogy hogyan különböztetjük meg egymástól a pozitív és negatív számokat. Erre szolgál az ún. előjelbit. A továbbiakban a számot negatívnak tekintjük, ha a bal szélső bit értéke 1, pozitívnak vesszük, ha 0. Az előző összeadásban a túlcsordult 1-es éppen az előjelbit helyén van. Az előjelbittel természetesen összeadási műveletet nem végzünk. Példaként ábrázoljuk a decimális 22 és -22 számokat 6 biten.

decimális alak	bináris kód	belső kód
22	10110	0 10110
-22	-10110	1 01010

(A -22 belső kódját a 22 kettes komplemenseként kaptuk.)

Kivonás

Példák:

a/

	bináris kivonás	bináris kivonás számítógéppel	
Kisebbítendő:	10001	10001	
Kivonandó:	- 01011	+ 10101	(a kivonandó 2-es
Különbség:	00110	100110	komplemense)
		↓ pozitív eredmény esetén az utolsó átvitelt törölni kell	

b/

	bináris kivonás	bináris kivonás számítógéppel	
Kisebbítendő:	101	101	
Kivonandó:	- 11011	+ 00101	(a kivonandó 2-es
Különbség:	10110 ↑	01010	komplemense)
		↓ 01010-nak a 2-es komplemense = 10110	

Ha az utolsó átvitel = 1, a kivonás eredménye pozitív, ha az utolsó átvitel = 0, akkor a kivonás eredménye negatív.

A bináris számok szorzása és osztása ugyanolyan lépések sorozatából áll, mint a decimális számok szorzása és osztása.

2. A számítógépes számábrázolás

2.1. Fixpontos számok

Fixpontos számtárolási formánál a szám kettes számrendszerbeli együtthatóit tároljuk helyi értékeiknek megfelelően egy rögzített nagyságú memóriaterületen. Ez leggyakrabban

- 1 bájt méretű,
- 2 bájt méretű szó (WORD),
- 4 bájt méretű duplaszó (DWORD).

A fixpontos számoknál fontos kérdés, hogy melyik pozíción helyezkedik el a szám egész- és törtrészét elválasztó jel, melyet a tizedesvessző analógiájára kettedes pontnak nevezhetünk.

A „kettedes pontról" kapta ez a számábrázolási forma az elnevezését, mivel az angolszáz szóhasználatban tizedesvessző helyett a tizedespontot alkalmazzák.

Minden számítógépnél a kettedes pontnak rögzített a helye, a gép a tárolt bitsorozatot ennek megfelelően értelmezi. A leginkább elterjedt a fixpontos számok egész számként való értelmezése, ebben az esetben a kettedes pont a legalacsonyabb helyi érték után helyezkedik el.

2^{15}	.	.	.	.	.	.	.	.	.	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

Értelmezése: 44_{10}

a kettedes pont helye

II.15. sz. ábra: Fixpontos egész szám (előjel nélküli szó)

Ha 8 bites a fixpontos szám, akkor ez a bináris számjegyek (0,1) 256 (2^8) kombinációját engedi meg, az ábrázolható számok tartománya tehát $00000000_2 = 0_{10}$-tól $11111111_2 = 255$-ig terjed.

A 8 bites adatok kezelhetők párban is, egy 16 bitből álló fixpontos szám például:

16 bites szó

1 0 0 1 1 1 0 0 0 1 0 0 1 0 1 1

a magasabb helyi értékű bitjei egy 16 bites szónak (1. bájt)	az alacsonyabb helyi értékű bitjei egy 16 bites szónak (0. bájt)

Ebben a szóban (előjel nélkül) a legkisebb ábrázolható érték 0, a legnagyobb pedig 2^{16}-1=65535.

A fixpontos számokkal a 2-es számrendszerben végezhetünk műveleteket. Például, két 16 bites szám összeadása a következő szerint történik.

```
  10011101    10000110
 +00101010    11010100
 ---------    --------
  11001000    01011010
```

A kivonást úgy végezzük, hogy a kivonandó 2-es komplemensét hozzáadjuk a kisebbítendőhöz.

Példa: $23A6_{16} - 124A_{16} = 115C_{16}$

Végezzük el a kivonást 16 bites memóriaszavakkal.

```
 1. bájt = 23_16   0. bájt = A6_16

 0 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0
 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 }  =  124A 2-es
                               1 }     komplemense
 -------------------------------
 0 0 0 1 0 0 0 1 0 1 0 1 1 1 0 0
    1       1       5       C
```

Összeadási példa előjel bittel (1 = negatív, 0 = pozitív)

$1A2C_{16} + (-0810_{16}) = 121C_{16}$

```
                  1. bájt          0. bájt
 1A2C_16   =  |0|0 0 1 1 0 1 0 0 0 1 0 1 1 0 0
 - 0810_16 =  |1|1 1 1 0 1 1 1 1 1 1 1 0 0 0 0
 =121C_16  =  |0|0 0 1 0 0 1 0 0 0 0 1 1 1 0 0
 előjel bit      1       2       1       C      16
```

A -0810_{16} előállítása:

0810_{16} =	0000 1000 0001 0000
A 0810_{16} 1'-es komplemense =	1111 0111 1110 1111
	+1
A 0810_{16} 2'-es komplemense =	1111 0111 1111 0000

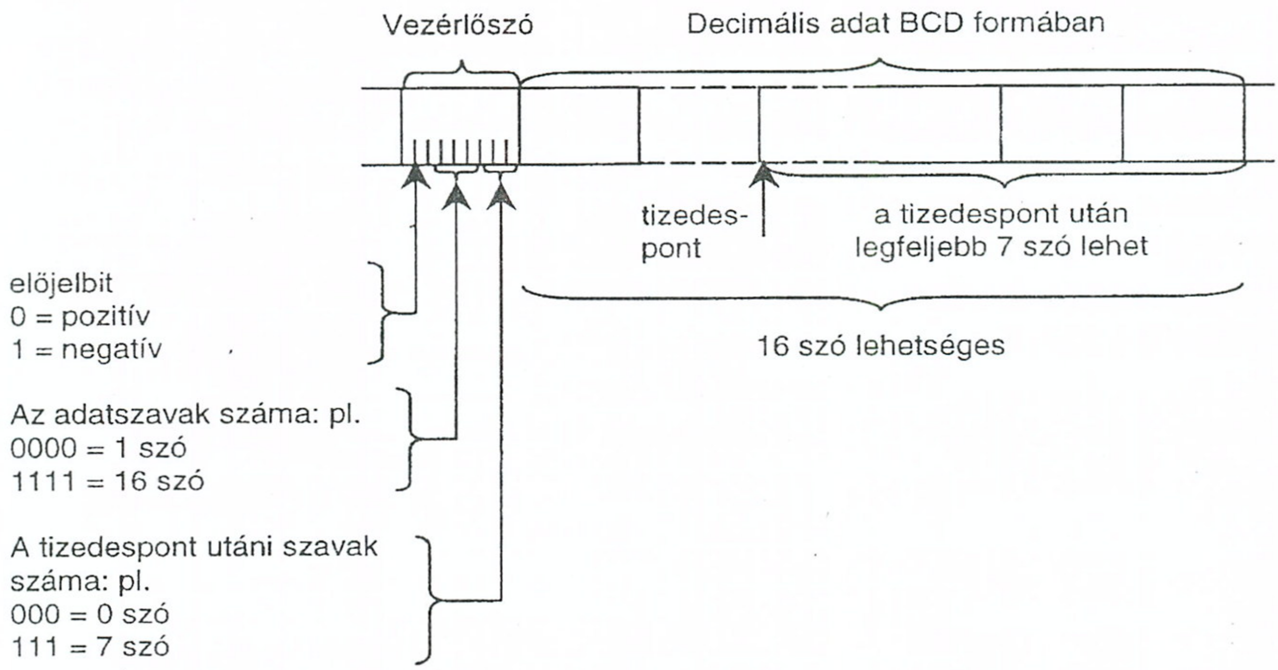
2.2. Binárisan kódolt decimális számok (BCD kód)

A BCD kódolás decimális számjegyek olyan bináris ábrázolási módszere, ahol minden decimális számjegyet helyi értékenként 4 bittel ábrázolnak. (Más szóval: a 0–9 számjegyeket 2-es számrendszerben fejezik ki, mindegyikben 4 helyi értéket véve.)

Bináris kód (szám)		**Decimális kód (szám)**
0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7
1000	=	8
1001	=	9

Előjeles BCD szám az eddigi szabályok szerint nem képezhető, ezért a BCD számok előjelének jelzésére speciális, úgynevezett vezérlőszó szükséges.

A vezérlőszó egyik lehetséges formátuma:



II.16. sz. ábra: Vezérlőszó formátum

BCD aritmetikai műveleteknél speciális szabályokat kell alkalmazni. Nézzük meg, milyen az **összeadás és a kivonás a binárisan kódolt decimális számokkal.**

Ennek lényege, hogy az egyes számjegyeknek megfelelő 4-bites egységeket összeadjuk vagy kivonjuk a 2-es számrendszer szabályai szerint, de az átviteleket a 10-es számrendszer szabályai szerint kell képezni a 4-bites egységek között.

Példák az összeadásra:

			29 + 34 = 63		87 + 79 = 166
Összeadandó	=	(29)	00101001	(87)	10000111
66_{16}	=		+ 01100110		+ 01100110
Összeg (1. lépés)	=		10001111		11101101
Összeadandó	=	(34)	+ 00110100	(79)	+ 01111001
Összeg (2. lépés)	=		11000011		01100110
C/IC	=		0 1		1 1
Faktor	=		10100000		00000000
2. lépés összege	=		+ 11000011		+ 01100110
Válasz	=		01100011		01100110
			6 3		6 6
C (2. lépés után)	=		0		1
Eredmény	=		63		166

Példák a kivonásra:

		75 - 21 = +54	25 - 71 = -46
Kisebbítendő	=	00100001	01110001
A kisebbítendő 2'-es komplemense	=	11011111	10001111
Kivonandó	=	+ 01110101	+ 00100101
Az 1. lépés összege	=	01010100	10110100
	=	1 1	0 1
	=	00000000	10100000
A 2. lépés összege	=	+ 01010100	+ 10110100
	=	01010100	01010100
		5 4	5 4
C (átvitel a 2. lépés után)	=	1	0
		1-es átvitel esetén az eredmény = 54	0-ás átvitel esetén az eredmény = 46 (100-54=46)

Az IBM kompatibilis PC-knél a BCD kódban ábrázolt számok 18 jegyű decimális egész számok, melyeket 10, egymást követő bájton tárolunk tömörítetten. A legfelső bájt legfelső bitje az előjel. Ha értéke 1, negatív a szám, ha 0, akkor pozitív. A legfelső bájt többi része kihasználatlan. A többi bájt egyenként két BCD jegyet tárol (II.15. sz. ábra).

79	78 72	71 64	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
S	0	dd	dd	dd	dd	dd	dd	dd	dd	dd

↑ előjel

d = egy 4 biten ábrázolt decimális számjegy

II.17. sz. ábra: BCD egész számok

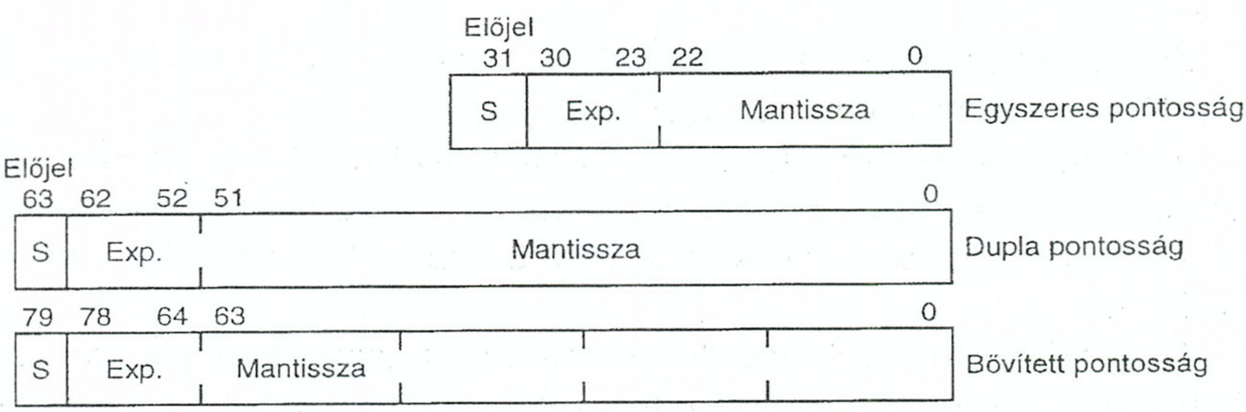
A formátum értéktartománya 0-tól ±999.999.999.999.999.999-ig tart.

2.3. Lebegőpontos számok

A hagyományos decimális lebegőpontos ábrázoláshoz (pl. $4{,}74 \times 10^3$) hasonló a bináris lebegőpontos ábrázolás. Az általános formátum: $\pm f \times 2^e$, ahol f egy bináris tört, a mantissza vagy törtrész, az e pedig a 2 hatványkitevője, exponense. Három mező alkot tehát egy lebegőpontos számot: az előjel, az exponens és a mantissza.

Tehát a lebegőpontos számok számítógépes ábrázolása során 2 fixpontos számot, a mantisszát, és az exponenst kell együtt kezelnünk.

A lebegőpontos számok pontos tárolási formátumát pontosságuk határozza meg. Leggyakrabban az egyszeres, dupla és bővített (kiterjesztett) pontosságú lebegőpontos számokat használjuk, ezek formátumát mutatja be a II.16. sz. ábra.



II.18. sz. ábra: Lebegőpontos adatformátumok a számábrázolás pontossága szerint

Az „S”-el jelölt előjel mező egyetlen bit, értéke 1, ha a lebegőpontos szám negatív; 0, ha a szám pozitív.

Az exponens mező a hatványkitevőt tartalmazza (Exp-el jelölve), ennek mérete:

- 8 bit egyszeres pontosságú számnál,
- 11 bit dupla pontosságú számnál,
- 15 bit bővített pontosságú számnál.

A mantissza mező tartalmazza a szám törtrészét. Ennek mérete:

- egyszeres pontosságú számnál 23 bit,
- dupla pontosságú számnál 52 bit,
- bővített pontosságú számnál 64 bit.

A lebegőpontos számok számítógépes ábrázolására nemzetközi szabványokat dolgoztak ki (ANS/IEEE 754 és 854-es szabvány). Ez egyrészt meghatározza a lebegőpontos számok gépi ábrázolásának adatformátumát, másrészt pontosan leírja a lebegőpontos számokkal végezhető műveletek algoritmusait.

3. Karakterek kódolása

A számítógépben szöveg és más, nem szám jellegű adat kezelésére binárisan kódolt karaktereket alkalmazunk.

Ehhez általában a következő karakterek szükségesek:

	26	kisbetű
	26	kisbetű
kb.	25	speciális jel
+	10	decimális szám
=	87	

Ennek alapján a karakterkészlet 87 karakterből áll, amelynek ábrázolására 7 bináris digit szükséges (2^7=128).

A számítógépekben általánosan **ASCII kódot** (American Standard Code for Information Interchange) alkalmazzuk az adatok ábrázolására. Az ASCII kódnál egy jelkombináció 8 bitből (1 bájtból) áll: 7 bit + 1 paritásbit.

A **paritásbit** a karakterkódot ellenőrző, hibajelző bitje. A paritásbit értéke 1, ha a karakterben a bitek között páros számú az 1-es bitek száma, és 0, ha ez a szám páratlan.

Paritásbitek felhasználásával egy karakter jelkombinációjában az 1-esek számát hibajelzés céljából egy rendszerre vonatkozóan tetszés szerint változtathatjuk vagy párossá, vagy páratlanná.

Paritás	Karakterek paritásbit		Az 1-es bitek száma
páratlan	1	0001100	3
	0	1110011	5
páros	0	0000000	0
	1	0001110	4

Példa: ASCII kóddal ábrázolt karakterek páros paritásellenőrzés esetén (II.19. sz. ábra):

hibás	E	n	r	e
0010000	11000101	11101110	01110010	01100101

Az ASCII kódolást a gyakorlatban kódtáblázatok segítségével hajtjuk végre (II.19. sz. ábra).

Például az '6' karakter kódja (függőleges + vízszintes)

decimális:	48 + 6 =	**54**
hexadecimális:	30 + 6 =	**36**
bináris:	00110000 + 0110 =	**00110110**

ASCII kódtábla

DECIMALIS			0	16	32	48	64	80	9[illegible]
	HEXADECIMALIS		00	10	20	30	40	50	6[illegible]
		BINARIS	0000 0000	0001 0000	0010 0000	0011 0000	0100 0000	0101 0000	0[illegible]
0	0	0000	NUL	DLE	SPACE	0	@	P	
1	1	0001	SOH	DC1	!	1	A	Q	
2	2	0010	STX	DC2	"	2	B	R	
3	3	0011	ETX	DC3	#	3	C	S	
4	4	0100	EOT	DC4	$	4	D	T	
5	5	0101	ENQ	NAK	%	5	E	U	
6	6	0110	ACK	SYN	&	6	F		
7	7	0111	BEL	ETB	'	7	G		
8	8	1000	BS	CAN	(	8			
9	9	1001	HT	EM		9			
[illegible]	A	1010	[illegible]						

II.19. sz. ábra: Az ASCII kódtábla alkalmazása

A számítógépek elterjedésével fogalmazódott meg az igény a kódtáblázat olyan kiegészítésére, mely a speciális nemzeti karakterek (például a magyar Á, É, Í stb.) kódolását is lehetővé teszi. Magyarországon az IBM által ajánlott ún. „852"-es kódlapot szabványosították, melyet a melléklet tartalmaz.

Ellenőrző kérdések

1. Mi a legfontosabb különbség a számológép és a számítógép működése között?
2. Határozza meg a számítógép fogalmát!
3. Milyen analógia alapján határozta meg Neumann János a számítógépek működési elvét?
4. Sorolja fel a Neumann elveket!
5. Fejtse ki a tárolt programozás elvének lényegét!
6. Milyen funkcionális egységekből épül fel a tárolt programozás elvén működő számítógép?
7. Mit jelent a kettes számrendszer alkalmazásának elve?
8. Mit jelent a soros utasításvégrehajtás elve?
9. Milyen tényezőkkel magyarázható a számítógépek gyors elterjedése és széleskörű felhasználhatósága?
10. Mit nevezünk rendszernek?
11. Hogyan függ a rendszer a vizsgálat céljától?
12. Milyen elemkapcsolatokat értünk a rendszer fogalmán?
13. Mit jelent az irányítás és a szervezés?
14. Mit nevezünk inputnak és outputnak?
15. Mit nevezünk algoritmusnak?
16. Mit nevezünk programnak és utasításnak?
17. Határozza meg az adat fogalmát!
18. Határozza meg az információ fogalmát!
19. Hogyan határozhatjuk meg az adat és az információ fogalmát a befogadó ember szerepe szerint?
20. Mit értünk információtechnológián?
21. Határozza meg az információs rendszer fogalmát!
22. Határozza meg az adatfeldolgozó rendszer fogalmát!
23. Milyen összefüggés van az adatfeldolgozó és az információs rendszer között?
24. Mit nevezünk informatikának?

25. Milyen összefüggés van az informatika és a számítástechnika között?
26. Mit értünk karakteren és milyen típusú karaktereket ismer?
27. Mit jelent a kódolás?
28. Mit nevezünk kódrendszernek?
29. Határozza meg a hardver fogalmát!
30. Határozza meg a szoftver fogalmát!
31. Mit értünk analóg adatokon, eszközökön?
32. Mikor nevezzük digitálisnak az adatokat, eljárásokat, az eszközök működését?
33. Mi jellemzi a digitális áramköröket?
34. Határozza meg a kombinációs és szekvenciális áramkör fogalmát!
35. Milyen részműveletekből áll az analóg–digitális átalakítás?
36. Mit értünk mintavétel alatt?
37. Mit értünk kvantálás alatt?
38. Hogyan működik a digitális–analóg átalakító?
39. Mi jellemzi a logikai áramköröket és milyen funkciókat látnak el?
40. Mi a bináris logika?
41. Mi a pozitív, illetve a negatív logikai konvenció?
42. Mi teremtette meg a mai digitális berendezések kialakításának elméleti alapjait?
43. Mi a kapcsolásalgebra?
44. Mit értünk Boole-műveleten?
45. Mivel határozzuk meg a Boole-műveleteket?
46. Mit nevezünk kapuáramkörnek?
47. MI jellemzi az „ÉS” kaput?
48. Mi jellemzi a „VAGY” kaput?
49. Mi jellemzi az „INVERTER” kaput?
50. Adja meg a NAND kapu igazságtáblázatát!
51. Adja meg a NOR kapu igazságtáblázatát!
52. Adja meg a „kizáró vagy” igazságtáblázatát!
53. Mit értünk kombinációs logikai hálózaton?
54. Hogyan lehet összeadó áramkört felépíteni?
55. Hogyan ábrázoljuk a számokat kettes számrendszerben? Mutasson erre két példát!
56. Hogyan ábrázoljuk a számokat tizenhatos számrendszerben? Mutasson erre két példát!
57. Számítsa át kettes számrendszerbe a 135 számot!

58. Számítsa át tízes számrendszerbe a következő számot:

 1 1 0 1 1 0 1 1

59. Adja össze a következő két számot kettes számrendszerben!

 0 1 0 1 1 1 0 1
 + 0 1 1 1 1 1 0 1

60. Mit nevezünk egy szám kettes komplemensének?
61. Mit nevezünk fixpontos számnak és milyen bájt méretben használjuk ezeket?
62. Mit értünk BCD kód alatt?
63. Adja meg a $123_{(10)}$ BCD kódját!
64. Mi a lényege a BCD kódú műveleteknek?
65. Hogyan ábrázoljuk a BCD kódú számok előjelét IBM kompatibilis PC-k esetén?
66. Milyen részekből épül fel a lebegőpontos szám?
67. Mit értünk egyszeres, dupla és bővített pontosságú lebegőpontos számon?
68. Mire használható és mit jelent az ASCII kód?
69. Mi a paritásbit?
70. Mennyi az ASCII kód jelkészlete és egy karaktert hány bittel kódolunk?

III. fejezet

A MIKROSZÁMÍTÓGÉP

1. Bevezetés

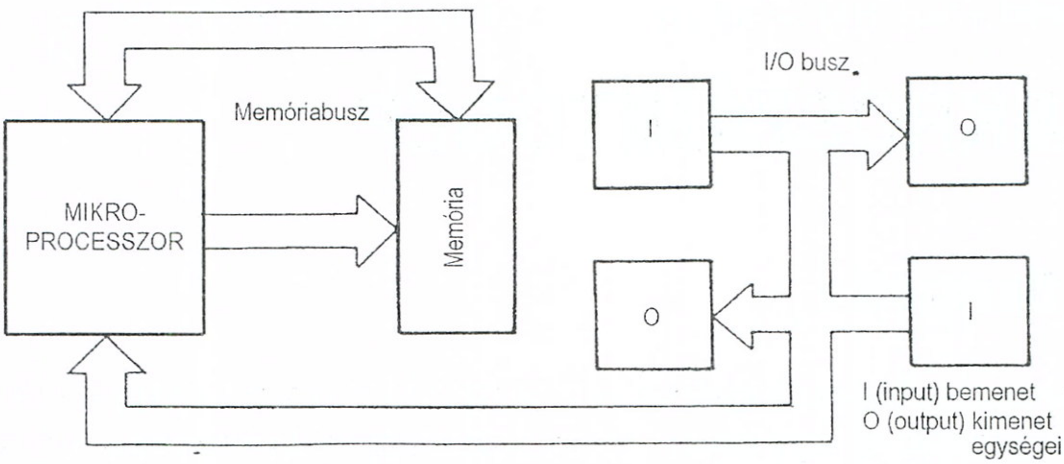
A mikroszámítógép fogalma a mikroprocesszorok megjelenését követően alakult ki egy olyan mikroprocesszort tartalmazó számítógépet jelölve, amely a kezdeti mikroprocesszorok teljesítménykorlátainak megfelelően kis teljesítményű és alacsony árfekvésű volt. Jelenleg, amikor a számítógépek egyre nagyobb hányada tartalmaz mikroprocesszort, a mikroszámítógépek elnevezést egyre kevésbé használják. Szerepét, megváltozott jelentéstartalommal a házi, a személyi, a professzionális számítógép, illetve a „mérnöki munkahely" fogalmak veszik át. (PC = Personal Computer, Workstation stb.)

2. A mikroszámítógép felépítése

2.1. A központi egység (CPU = Central Processor Unit)

! Minden mikroszámítógépnek négy alapvető funkcionális egysége van:

- A **bemeneti egység**, amely az adatok és a program bevitelét biztosítja.
- A **főtár (memória)**, amely a műveletek elvégzéséhez szükséges adatokat és programokat, valamint az eredményt tárolja későbbi felhasználás céljából.
- A **mikroprocesszor**, amely a memóriából kapott adatokon a programnak megfelelő logikai és számítási műveleteket elvégzi.
- A **kimeneti egység**, amelyen keresztül az eredmény eljut a felhasználóhoz.



III.1. sz. ábra: Mikroszámítógép alapegységei

! A funkcionális egységek közötti kommunikációs kapcsolatokat, azaz a címek, adatok, vezérlőinformációk átvitelét, a mikroszámítógép busz vagy sínrendszere biztosítja.

A busz egy vezetékrendszer, melynek fontosabb fajtái:

- a belső busz, mely a mikroprocesszoron belüli részegységeket köti össze,
- a memóriabusz, mely a processzort köti össze a tárral,
- az I/O vagy rendszerbusz, mely a mikroszámítógép processzorát a be/kiviteli egységekkel kapcsolja össze.

A mikroszámítógép processzorát és főtárolóját a sínrendszerrel együtt a számítógép **alaplapján** szokták elhelyezni.

2.2. Perifériák

A perifériák a számítógépnek adatbevitelre, illetve -kivitelre vagy átmeneti külső adat- és programtárolásra szolgáló egységei.

Ezek tipikusan a perifériavezérlő egységeken keresztül kapcsolódnak a központi egység rendszerbuszához.

A leggyakrabban használt perifériatípusok:

Beviteli perifériák:
- billentyűzet,
- egér,
- fényceruza,
- optikai letapogató (scanner).

Kiviteli perifériák:
- monitor (display),
- nyomtatók,
- rajzolók,
- akusztikus kiviteli eszközök.

Adattárolásra és bevitelre/kivitelre is használt perifériák:
- mágneslemezes tár (merevlemezes – Winchester, vagy hajlékonylemezes – floppydiszk),
- mágnesszalagos tár (sztrímer),
- optikai lemezes tár (CD-ROM, DVD-ROM, DVD-RAM).

3. A mikroszámítógép működése

3.1. Órajel és gépi ciklus

A mikroszámítógép folyamatos működését periodikusan kiadott jelek – órajelek – biztosítják, amelyek egyrészt az ütemnek megfelelően engedélyezik az adatok jeleinek belépését az áramkörökbe, másrészt szinkronizálják az áramkörök állapotváltozásait. Ezekkel az órajelekkel találkozhatunk akkor, ha egy processzorról például azt olvassuk egy hirdetésben, hogy 400 MHz-es.

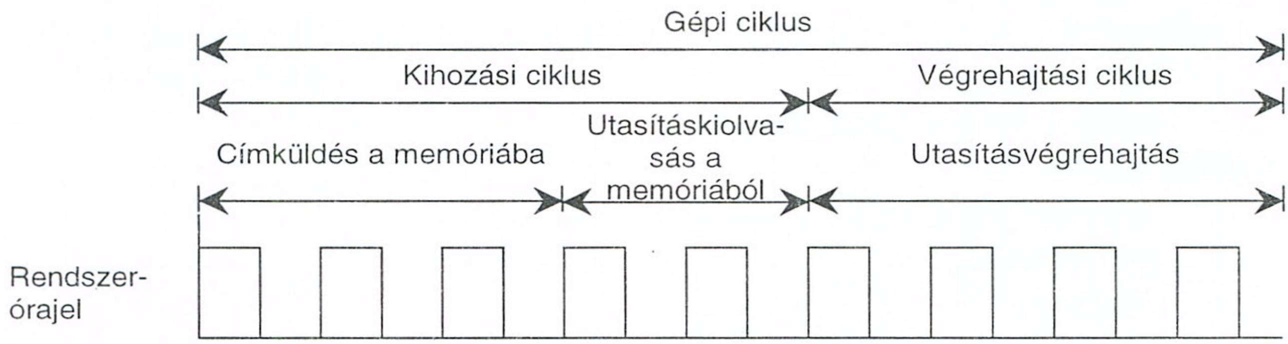
Egy gépi utasítás végrehajtása általában több óraciklus alatt megy végbe.

A gépi ciklus az az **időtartam**, amely egy processzor-alapművelet végrehajtásához szükséges.

Egy processzor-alapművelet: a belső tárban tárolt adatok kiküldése (kikapuzása) a belső buszokra, a kiválasztott alapművelet (pl. összeadás, logikai művelet) végrehajtása a processzor műveletvégző egységeiben és az eredmények beküldése (bekapuzása) – a buszon keresztül – a kijelölt belső tárolóba.

Egy gépi ciklus tehát két fázisból áll (III.2. sz. ábra.):

- az utasítás-kihozási ciklus, mely alatt (az utasításszámláló által) meghatározott tárolóhelyről a processzor kiolvassa az utasítást;
- az utasítás végrehajtásának ciklusa.



III.2. sz. ábra: A gépi ciklus fázisai

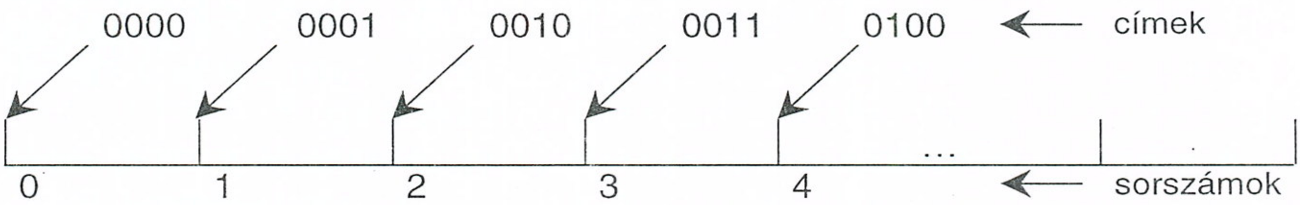
3.2. Tárak, tárolók (memória)

A számítógép az adatok és az utasítások bitjeit egy, az adott gépre jellemző rekeszekben tárolja. A rekeszeket felépítő bitek száma a mikroszámítógép típusára jellemző állandó szám. A legelterjedtebb a bájt, ez 8 bitből áll.

Tár minden olyan eszköz, amely adatokat őriz meg a későbbi kiolvasás céljára. A tár az elektronikus berendezéseknek az a része, amelyben az adatok, a program és az eredmény tárolása történik, általában elektromos vagy mágneses működés alapján.

Az adatok beírása és kiolvasása a tárolókból a címezhetőség elve alapján történik. Ez azt jelenti, hogy minden egyes rekesznek (bájtnak) van egy sorszáma, mely a gépi utasításokban a rekeszt egyértelműen azonosítja.

A rekeszek vagy bájtok sorszámát bináris formában a rekesz vagy bájt abszolút (vagy fizikai, illetve lineáris) címének nevezzük. (A sorszámozást 0-tól kezdjük.)



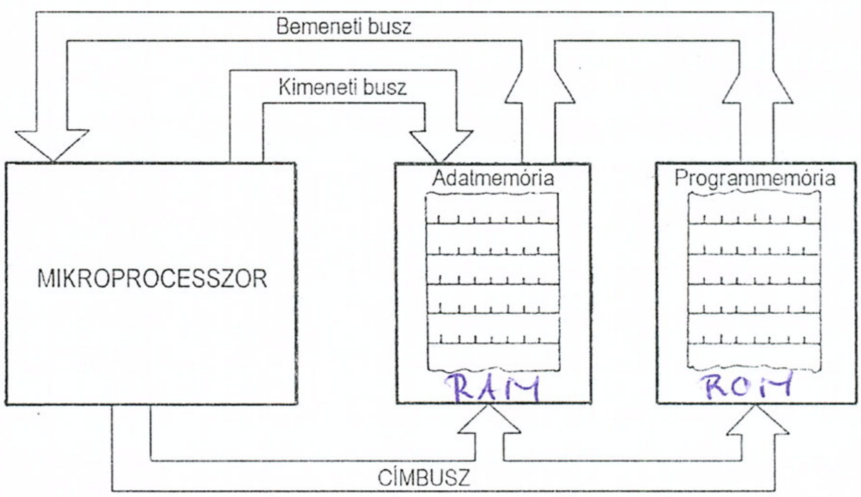
III.3. sz. ábra: A tároló rekeszek címei

A tárkapacitást a tárolható rekeszek, bájtok vagy bitek számával mérik.

A legfontosabb mértékegységek:

- a Kbájt (kilobájt), mely 2^{10} = 1024 bájtot jelent (például 4 Kbájt = $2^{10} \cdot 4$ bájt = 4096 bájt)
- a Mbájt (Megabájt), mely 2^{20} bájtot jelent.

A tárolókat a buszrendszerrel csatlakoztathatjuk a processzorhoz. Ennek elvi megoldását mutatja be a III.4. sz. ábra.



III.4. sz. ábra: A tárak csatlakoztatása a processzorhoz

A különböző tárolókat

- az adatok elérése,
- az adatok átírhatósága,
- a tárolók funkciója
- és fizikai működési elvük

szerint osztályozhatjuk.

Az adatok elérése szempontjából megkülönböztetünk

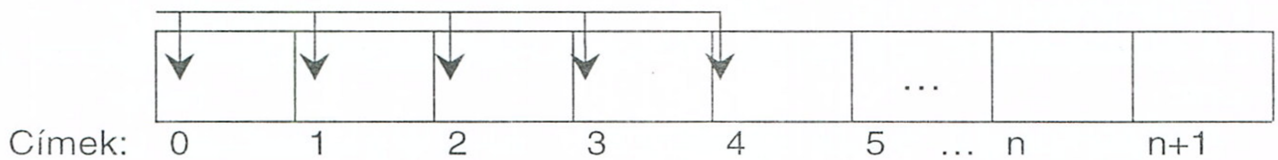
- soros és

- közvetlen hozzáférésű

tárolókat.

A **soros tárolóban** a tárolóhelyek **időben egymást követően** válnak hozzáférhetővé az információ beírásakor és kivitelekor. Egy rekesz hozzáférési ideje függ az utoljára hivatkozott (kikeresett) rekesz és a keresett rekesz viszonylagos címeitől. Erre példa a hetvenes évek fő háttérmemória típusa, a **mágnesszalag**.

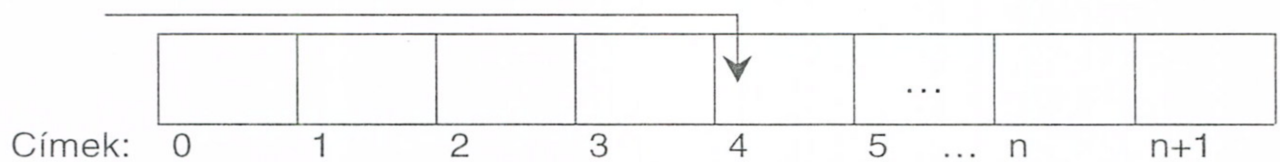
A soros hozzáférésű tárolónál, ha ki akarjuk olvasni pl. a 4. című adatot, akkor sorba végig kell olvasni a 0, 1, 2, 3 című adatokat is.



III.5. sz. ábra: A soros hozzáférésű tároló működése

A **közvetlen hozzáférésű tárolóban** az adatokat felvitelük sorrendjétől függetlenül lehet elérni, bármelyik tárolt adat **beírása körülbelül azonos idő alatt** történik meg, valamennyi rekesz hozzáférési ideje lényegében egyenlő. Közvetlen hozzáférésű tárolóra példa a **mágneslemez** (a Winchester, vagy a floppydiszk).

A közvetlen hozzáférésű tárolóban azonnal kiolvashatjuk a 4. című adatot, az előtte elhelyezett adatoktól függetlenül.



III.6. sz. ábra: A közvetlen hozzáférésű tároló működése

Az asszociatív tároló a tartalma alapján címezhető. Az asszociatív tárolóban tehát a tároló bemenetére egy konkrét adatértéket írunk, és a tároló kimenetén megkapjuk, hogy ezt az adatértéket a tároló tartalmazza-e vagy sem. Vagyis ez olyan tárolótípus, melynél a rekeszekben tárolt információ nem a rekeszek címei alapján, hanem a tárolt információ egy része (keresési kulcs) alapján olvasható ki.

Az asszociatív tár **gyors** működésű, a **keresési kulcs összehasonlítása** a rekeszekben tárolt tartalommal az **összes rekeszre nézve egyidejűleg történik meg.** Ez természetesen csak úgy való-

sítható meg, hogy az asszociatív tár minden rekeszhez külön-külön összehasonlító áramköröket tartalmaz. Ezért ez a tárolótípus a többihez képest lényegesen drágább.

A tárban tárolt adatok átírhatósága szempontjából megkülönböztetünk:

- csak olvasható tárakat (ROM = Read Only Memory)
- írható-olvasható (programozható) tárakat (RAM = Random Access Memory)

Az adatok rögzítése – beírása – a memóriában lehet **végleges** (fix memóriák, például: ROM), vagy **módosítható** (például: RAM, EP-ROM).

A **ROM** típusú memóriából az egyszer – általában technológiai módszerekkel – beírt információ csak kiolvasható, tartalma nem változtatható meg.

Az **írható/olvasható tárolók** (RAM) tartalma módosítható, azaz író üzemmódban a külső adatokkal azonossá tehető a memóriarekeszek tartalma.

Az írható/olvasható tárolók lehetnek **dinamikus** vagy **statikus** működésűek. A dinamikus tároló a beírt adatokat csak periodikus jelregenerálással képes tetszés szerinti ideig tárolni. A statikus tároló a beírt adatokat elvileg korlátlan ideig megőrzi. Természetesen mindkét eset folyamatos áramellátást feltételez, a tápfeszültség kimaradása esetén mind a dinamikus, mind a statikus tároló tartalmát elveszti.

Az **újraprogramozható** „csak olvasható" **tárak** a számítógép üzemszerű működése során csak olvashatóak. Ugyanakkor speciális eszközökkel (például EPROM esetében ultraibolya fénnyel) ezeknek a tárolóknak a tartalma törölhető, és ezt követően új adatokkal feltölthető.

A feladataik szerint a tárolók lehetnek:

- **operatív tárak** (belső tár, főtár, központi tár, kissé pontatlanul RAM) és
- **háttértárak** (külső tárak).

Az operatív tárban tároljuk azt a programot, melyet a számítógép aktuálisan végrehajt. A többi programot a háttértárakon helyezzük el és akkor olvassuk be az operatív tárba, ha a megfelelő programot végre akarjuk hajtatni. (Ezt a program futtatásának is nevezik.)

! Működési elvük, megvalósításuk alapján az alábbi főbb tártípusok különböztethetők meg:

- mágneslemezes tár (floppydiszk, Winchester),
- mágnesszalagos tár,
- félvezető tár,
- optikai tár.

! **Háttértár**ként jelenleg még döntően a mágneslemezes tárak használatosak, de terjedőben vannak az optikai tárak is (csak olvasható ill. írható/olvasható tárként).

Archív tárként, mentési céllal speciálisan kialakított mágnesszalagos tárak (sztrímerek) használhatóak előnyösen.

A félvezető tárolók alapeleme a két stabil állapotú billenőkör a flip-flop, amely 1 bit információ tárolására alkalmas. A tárolt bit értéke lehet "0" vagy "1". Állapotát csak külső jel hatására változtatja meg.

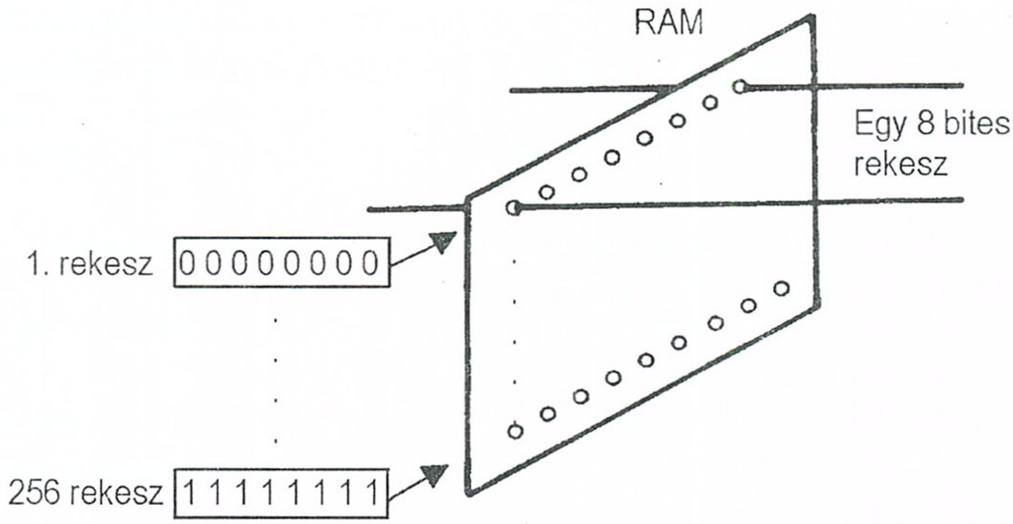
3.3. Memóriacímzés

Az adatokat a memóriából kiolvashatóvá, vagyis kikereshetővé, és ezért **megcímezhetővé** kell tenni. A számítógépeken a legkisebb címezhető adategység (amit a memóriában elérhetünk) – mint már említettük – a rekesz, ami általában egy bájt. Ennek megfelelően minden bájtnak saját címe van, ez a memórián belül a bájt sorszáma (az első bájtnak van a legalacsonyabb címe: zérus, az utolsó bájtnak van a legmagasabb címe). A legmagasabb cím értékét a memória mérete határozza meg.

! A tárolóknak azt a részét, melyet a tárolás, az átvitel, a feldolgozás szempontjából a számítógép egy egységként kezel, **szónak** nevezzük. Ez lehet 1, 2, 4, 8 bájt, azaz 8, 16, 32 vagy 64 bitből állhat.

A következő példákban a memória címzését, az adatok kiolvasását, valamint a gépi utasítás végrehajtását leegyszerűsítve mutatjuk be (például 16 bites memóriacímek), koncentrálva az eljárás lényegére. Természetesen ez a mai számítógépekben jóval bonyolultabb módon történik. Ennek ellenére az eljárás elve napjainkban is lényegében azonos.

Először – az egyszerűség kedvéért – tételezzük fel, hogy a memória egy chipből álló részekből, vagyis egy chipes memóriamodulokból épül fel (III.7. ábra).



III.7. sz. ábra: Egy chipből álló memóriamodul

Ekkor chipen belüli rekeszek címzéséhez szükséges címbitek száma a chip méretétől függ. Például a III.7. ábrán látható 256 memóriarekesz kapacitás esetén a rekesz címe 8 bitből áll.

Különböző számrendszerekben ábrázolva:

- a legkisebb cím = $0000\ 0000_2 = 00_{16} = 00_{10}$
- a legnagyobb cím = $1111\ 1111_2 = FF_{16} = 255_{10}$

Egy írható/olvasható memóriamodul általában több chipből áll. Ezért minden memóriacím tartalmazza egyrészt a szükséges chipek, másrészt az aktuális rekesz kiválasztására vonatkozó adatokat. A chipkiválasztó bitek kiválasztják a memórián belül azt a memóriamodult, amely azokat a chipeket tartalmazza, amelyben az adatot tárolják. A rekesz kiválasztására szolgáló címbitek pedig egy-egy rekeszt azonosítanak a memóriamodulon belül.

Például, ha egy mikroszámítógép 65536_{10} (2^{16}) memóriarekesz címzésére alkalmas, akkor 16 bit kell a legnagyobb memóriacím kifejezéséhez ($FFFF_{16}$)

Ha a memóriachip 1024 bites, akkor egy rekesz címzéséhez 10 bitet kell felhasználni, így a chipek kiválasztására 6 bitünk maradt, ezzel (2^6) 64 memóriamodult lehet kiválasztani.

Így egy 16 bites memóriacím a következő részekből áll:

C C C C C C S S S S S S S S S S

6 db chip-kiválasztó bit (modulcím)	10 db rekeszcímző bit

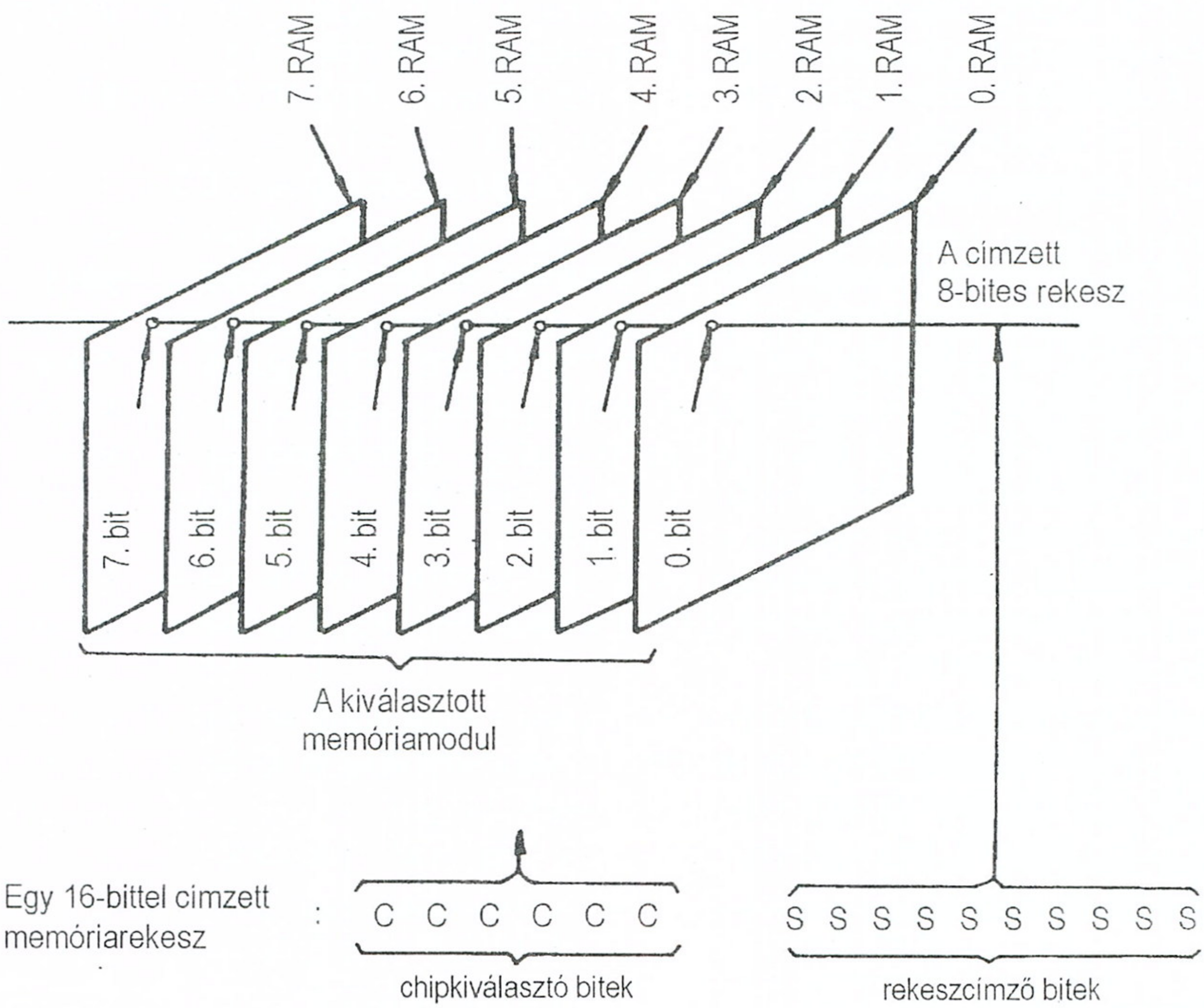
A tervezőtől függ az, hogy a címzéshez rendelkezésre álló bitek hogyan oszlanak meg a rekeszek és a chipeket tartalmazó memóriamodulok címzése között.

Egy konkrét memóriatípus esetén a következő döntéseket kell hozni:

- egy chip hány bitből áll, ez meghatározza a rekeszek címzéséhez szükséges bitek számát,
- egy rekesz hány bitből áll, ez meghatározza hogy egy memóriamodult hány chipből kell felépíteni,
- a memória maximálisan hány memóriamodult tartalmazhat. Ez meghatározza a modulok címzéséhez szükséges chipkiválasztó bitek számát.

Ha egy rekesz 8 bitből, azaz 1 bájtból áll (lásd III.8. ábra), akkor az előző példában bemutatott memória maximális kapacitása

$$8 \cdot 1024 \cdot 64 \text{ bit} = 64 \text{ Kbájt}.$$



III.8. sz. ábra: Egy 16 bittel címzett rekesz azonosítása a memóriában

Példa: **címzés 16 bittel.**

A rekesz címe: 0110101110_2 = $1AE_{16}$
A chip címe : 000111_2 = 07_{16}

	chip	rekesz	
	07_{16}	$1AE_{16}$	
A teljes cím :	0 0 0 1 1 1	0 1 1 0 1 0 1 1 1 0	= $1DAE_{16}$

0 0 0 1	1 1 0 1	1 0 1 0	1 1 1 0
1	D	A	E

3.4. A gépi utasítások végrehajtásának lépései

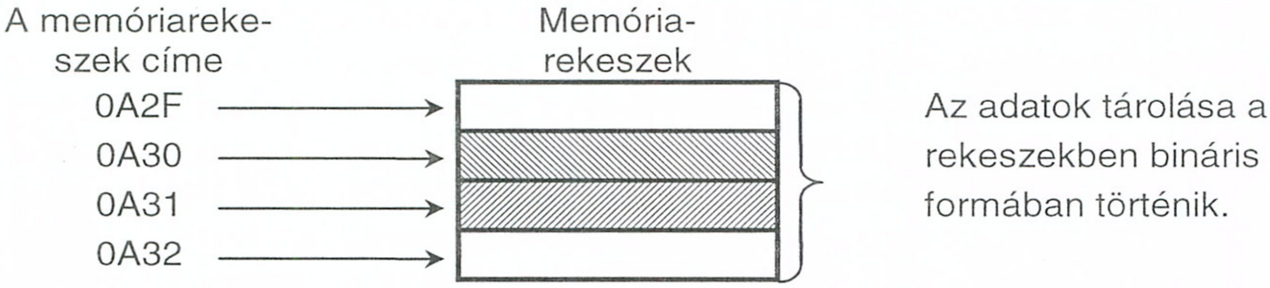
Egy, a processzor által végrehajtható számítógépes program gépi utasítások sorozatából áll, melyeket bináris formában tárolunk a memóriában. Minden gépi utasítást – összeadás, szorzás, vezérlésátadás stb. – további elemi lépésekre bontva hajt végre a számítógép.

Vizsgáljuk meg, hogy például egy bináris összeadás elvégzéséhez milyen elemi lépések végrehajtása szükséges.

Legyen a feladat az, hogy a memória $0A30_{16}$ címén tárolt számhoz hozzá kell adni a $0A31_{16}$ címén tárolt számot, és az eredményt a $0A30_{16}$ címen kell tárolni.

Ha a memóriából az összeadásnak megfelelő gépi utasítást már kiolvasta és értelmezte a processzor, akkor a feladat végrehajtásához a következő lépések szükségesek:

1. Az utasításban lévő cím alapján azonosítani kell azt a memóriarekeszcímet, amelyen az első összeadandó van
2. Az azonosított memóriarekesz tartalmát át kell vinni a mikroprocesszorba.
3. Azonosítani kell azt a memóriarekeszcímet, amelyen a második összeadandó tárolva van.
4. A második lépésben átvitt szóhoz hozzá kell adni a harmadik lépésben kikeresett szót.
5. Azonosítani kell annak a memóriarekesznek a címét, amelyben az összeget tárolni fogjuk.
6. Át kell vinni az összeadási művelet eredményét az ötödik lépésben kikeresett memóriarekeszbe.



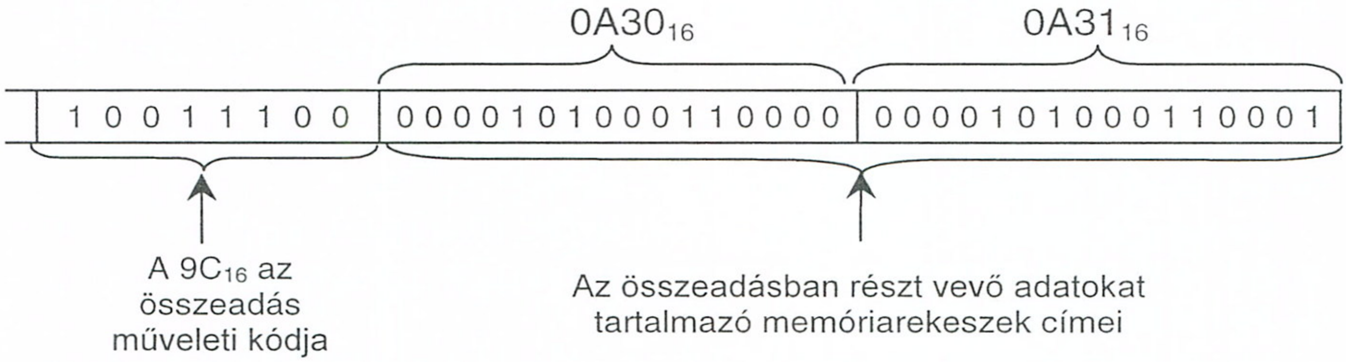
III. 9. sz. ábra: Az adatok elhelyezkedése a memóriában

Az összeadáshoz tehát hat elemi lépés szükséges.

A példánkban az összeadás gépi utasításkódja 5 bájtot foglal el (lásd III.10. sz. ábra).

Ez tartalmazza

- a műveleti kódot (Mit kell végrehajtani?),
- az adatok címeit a memóriában (Mivel kell a műveletet elvégezni?).



III.10. sz. ábra. Az összeadás gépi utasításkódjának ábrázolása

Példánk annyiban általánosítható, hogy a számítógép gépi utasítása mindig tartalmaz egy műveletikód részt (mely az elvégzendő műveletet azonosítja) és egy címrészt (mely a műveletben részt vevő adatok címét tartalmazza).

4. A mikroszámítógép szoftvere

Az előző fejezetben láttuk, hogy a számítógép a számára bináris formában megadott gépi kódú utasításokat képes csak végrehajtani.

A számítógépprogram **bináris számsorozat** alakjában is megírható, de ekkor igen nagy az olyan hibák valószínűsége (például 0 vagy 1 felcserélése), amelyek igen nehezen találhatók meg.

A programot hexadecimális számjegyekkel írva kevesebb a lehetőség hiba elkövetésére, mivel négy bináris számjegy (négyjegyű bináris szám) megadható egy hexadecimális számjeggyel.

A hexadecimális számjegyekkel megjelenített programban a hibák is könnyebben megtalálhatók. Egyszerűbb egy rossz helyre írt hexadecimális számjegyet észrevenni, mint egy szemkápráztató bináris mintában a téves 0 vagy 1-et felismerni.

Végül azonban a programot át kell alakítani bináris jelsorozattá, mivel a memóriában így tárolják és a processzor csak ebben a formában képes a program végrehajtására.

Az első számítógépeket (ezek még nem voltak a mai értelemben vett számítógépek) még áramköri szinten a vezetékek „dugaszolásával" programozták, később a gépi kódú utasításokat bináris formában lyukszalagra kellett rögzíteni. (A lyukszalagon egy lyuk léte vagy nemléte felelt meg a bináris 1 vagy 0 jelnek.) Ennek során némi könnyítést jelentett, hogy a lyukasztógép klaviatúráján nyolcas számrendszerben (3 bit egy nyolcas számrendszerbeli számjegy) kellett a számokat beütni.

A számítógépek elterjedésével egyre nyilvánvalóbbá vált, hogy ez a módszer nem elég hatékony. Először az ún. **assembly programozási** nyelv alakult ki, ahol már a műveleti kódokat az értelmüknek megfelelő angol szavak rövidítésével jelölték (mnemonik) és a gépi utasításban lévő címeknek is lehetett ún. szimbolikus nevet (címet) adni.

Így például az előző fejezetben bemutatott összeadás bináris kódja helyett a programozók az utasításokat szimbolikus formában fogalmazhatták meg.

gépi kód	1 0 0 1 1 1 0 0	0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0	0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 1
assembly nyelv	ADD	ADAT1, ADAT2	

III.11. sz. ábra: Összeadási utasítás assembly nyelven és gépi kódban

Felmerül a kérdés, hogy a szimbolikus műveleti kódot és címeket hogyan fogja megérteni a számítógép, ha a processzor csak a bitek nyelvén ért?

A válasz (ha már rájöttünk a megoldásra) tulajdonképpen egyszerű: egy olyan programot kell készítenünk, mely ha beolvassa az ADD betűknek megfelelő karakterek bináris értékét, akkor ehhez mindig az összeadás bináris műveleti kódját rendeli hozzá és ezt a tárolóban letárolja, és ehhez hasonlóan jár el az ADAT1, ADAT2 címek esetében is.

Azokat a programokat, melyek a szimbolikus programnyelveken megfogalmazott utasításokból a processzor számára gépi kódként értelmezhető bitsorozatot állítanak elő, **fordítóprogramok**nak nevezzük.

Tehát a III.10. ábrán bemutatott bináris formában tárolt gépi kódú utasítást a fordítóprogram állítja elő, a

$$\text{ADD} \Rightarrow 9C_{16} = 10011100$$
$$\text{ADAT1} \Rightarrow 0A30_{16} = 0000101000110000$$
$$\text{ADAT2} \Rightarrow 0A31_{16} = 0000101000110001$$

hozzárendelések segítségével.

A szimbolikus programnyelvek segítségével felgyorsult a programozás és az emberek egyre több feladatot fordítottak le a számítógépek nyelvére. Egyre jobban kihasználták a számítógépnek azt a képességét is, hogy a programokat lehetőség van a számítógép tárolóeszközein megőrizni, és szükség esetén ismételten felhasználni. Ha például valaki egy másodfokú egyenlet megoldását már beprogramozta és ezt tárolták, ezt bárki átvehette, végrehajtathatta a számítógéppel vagy saját programjába beépíthette.

Így a számítógépet működtető programok összességében, vagyis a **szoftver**ben évről-évre egyre több szellemi érték halmozódott fel.

Az 1950-es évektől kezdve egyre-másra jöttek létre az emberi nyelvhez egyre inkább közelítő, ún. **magasszintű programnyelvek**. (FORTRAN 1954, ALGOL 60, COBOL 1959, BASIC 1964, PASCAL 1968, C 1974), melyekkel már sokkal könnyebben lehetett gyorsan áttekinthető és hibamentes programokat írni. Természetesen ennek az ára az volt – mivel a magasszintű programnyelveknél egy utasítás általában már több gépi kódú utasításra lesz lefordítva –, hogy a fordítóprogramok is egyre bonyolultabbá váltak.

Az egyre több feladat megoldására képes szoftver egyes részei a fejlődés során a programok felhasználási célja szerint is kezdtek elkülönülni.

Így alakult ki a gép alapműködtetését a felhasználás céljától függetlenül biztosító, általános (például egy eszköz hibás működésének behatárolását, erről a felhasználó tájékoztatását is elvégző) programok együttese, az **operációs rendszer**.

Ezt a számítógépet gyártó cég általában a géppel együtt szállította.

Az operációs rendszerek a fejlődés során egyre több segédprogrammal (utility) egészültek ki (másolók, rendezők stb.), melyek tovább segítették a számítógépet felhasználók munkáját.

Még az 1970-es években is jellemző volt, hogy ha egy szervezet egy meghatározott célra egy számítógépet vásárolt, akkor az adott feladatnak megfelelő programok elkészítését is vállalnia kellett.

Az egy-egy konkrét alkalmazás céljára kifejlesztett programokat **felhasználói programok**nak nevezzük.

A felhasználói programok fejlesztésében az idők folyamán célszerű munkamegosztás alakult ki az egyes speciális szoftver szakterületekre szakosodott cégek és a számítógépet alkalmazó szervezetek között. Nyilvánvalóan gazdaságosabb volt, ha az egyes sajátos, több felhasználónál is azonos módon jelentkező feladatokat megoldó szoftvereket erre szakosodott **szoftverházak** fejlesztik ki és gondozzák.

Erre az egyik legjellemzőbb példa az **adatbáziskezelő** szoftverek kialakulása, melyekkel egységes módon meg lehetett oldani a felhasználók adattárolással és visszakereséssel kapcsolatos feladatait.

A felhasználás szakterülete szerint specializálódott szoftverekre néhány további példa:

- irodaautomatizálási (office) szoftverek,
- mérnöki tervezőrendszerek (CAD),
- elektronikus levelező szoftverek stb.

Az eddigieket összefoglalva a mikroszámítógépet működőképessé tevő szoftverek fontosabb típusai a következők:

- az operációs rendszer és segédprogramjai,
- az adatbáziskezelő rendszer,
- a programnyelvek fordító és könyvtárkezelő programjai,
- az általánosan használt gyári programcsomagok (szövegszerkesztők, elektronikus levelezőprogramok stb.),
- speciális célú gyári programcsomagok (pl. a számítógépes tervező, CAD rendszer),
- a felhasználó által írt programok.

Ellenőrző kérdések

1. Milyen funkcionális részekből épül fel a mikroszámítógép?
2. Mi a feladata a be/kimeneti egységeknek?
3. Mi a feladata a mikroprocesszornak és a tárnak?
4. Mivel lehet biztosítani a kommunikációs kapcsolatokat a mikroszámítógép funkcionális egységei között?

5. Milyen részegységeket tartalmaz a mikroszámítógépek alaplapja?
6. Mit nevezünk perifériának?
7. Sorolja fel a beviteli perifériákat!
8. Sorolja fel a kiviteli perifériákat!
9. Sorolja fel az adattárolásra használt perifériákat!
10. Mit biztosítanak az órajelek?
11. Mi a gépi ciklus és milyen részekből áll?
12. Mit nevezünk tárnak?
13. Mit nevezünk címnek?
14. Mit nevezünk bájtnak, Kbájtnak és Mbájtnak?
15. Milyen szempontok szerint osztályozhatjuk a mikroszámítógépek tárolóit?
16. Mi jellemzi a soros tárolóeszközök működését?
17. Mi jellemzi a közvetlen hozzáférésű tárolókat?
18. Hogyan működik az asszociatív tár?
19. Milyen tárolókat különböztetünk meg a tárolt adatok átírhatóságát figyelembe véve?
20. Mi a ROM és a RAM?
21. Mi a különbség a statikus és a dinamikus tárak között?
22. Mi a feladata az operatív tárnak?
23. Milyen elven működő tárolótípust használunk háttértárként?
24. Mire használhatók a sztrímerek?
25. Mi határozza meg a tárolóban a legkisebb és legnagyobb címet?
26. Milyen részekből épül fel egy memóriacím?
27. Mit nevezünk memóriamodulnak?
28. 1024 bites memóriachip esetében hány bitet kell felhasználni a memóriarekesz címzéséhez?
29. Miből áll a processzor által végrehajtható program?
30. Milyen elemi lépésekből áll egy gépi utasítású összeadás végrehajtása?
31. Milyen részekből áll egy gépi kódú utasítás?
32. Mi a különbség a gépi kódú és az Assembly nyelvű utasítás között?
33. Mit nevezünk fordítóprogramnak?
34. Mi jellemzi a magasszintű programnyelveket?
35. Mit értünk operációs rendszeren?
36. Mit nevezünk felhasználói programnak?
37. Sorolja fel a mikroszámítógépes szoftverek fontosabb típusait!

IV. fejezet

HARDVER ALAPISMERETEK

A. MIKROPROCESSZOR

1. Bevezetés

A legtöbb egychipes mikroprocesszort MOS-technológiával állítják elő, amivel a kívánt nagy bonyolultsági fok (LSI, VLSI) elérhető. Cél, hogy egyetlen chipen belül minél komplexebb processzor legyen megvalósítható. !

2. A mikroprocesszor feladata

A mikroprocesszor olyan nagy bonyolultságú (LSI, sőt VLSI) félvezető eszköz, amely a digitális számítógép központi egységének (CPU) a feladatait végzi el. A mikroprocesszor dekódolja és végrehajtja az utasításokat, vezérli a műveletek elvégzéséhez szükséges belső adatforgalmat és a csatlakozó perifériális berendezések tevékenységét. !

3. A mikroprocesszor funkcionális egységei

A mikroprocesszoroknak három alapvető funkcionális blokkjuk van:

- a regiszterek,
- az aritmetikai logikai egység (ALU = Arithmetical Logical Unit),
- a vezérlőegység.

Ezeket a processzor belső buszrendszere köti össze.

3.1. Regiszterek

Egy CPU általában több gyors működésű tárolót, úgynevezett regisztert tartalmaz abból a célból, hogy a memóriából kihozott adatokat tárolja. Ezek tehát gyors írható-olvasható munkatárak. A különböző regiszterek meghatározott feladatokhoz vannak hozzárendelve. Ezek elvégzése közben a szükséges adatokat átmenetileg – rövid ideig – a regiszterekben tároljuk.

Egy regiszter tárolókapacitása többnyire egy szó, a regiszterek hosszát a benne tárolható bitek, bájtok vagy szavak számával mérik.

A regiszterek egy része szigorúan meghatározott feladatokhoz van hozzárendelve, és a felhasználói programok által nem használhatók. Ezeket **rendszerregiszterek**nek nevezzük.

A rendszer (speciális funkciójú) regiszterek mellett még adattárolási, indexelési, címzési stb. célokra használatosak az ún. **általános** (univerzális) **regiszterek**. Ezeket a felhasználói programok is alkalmazhatják.

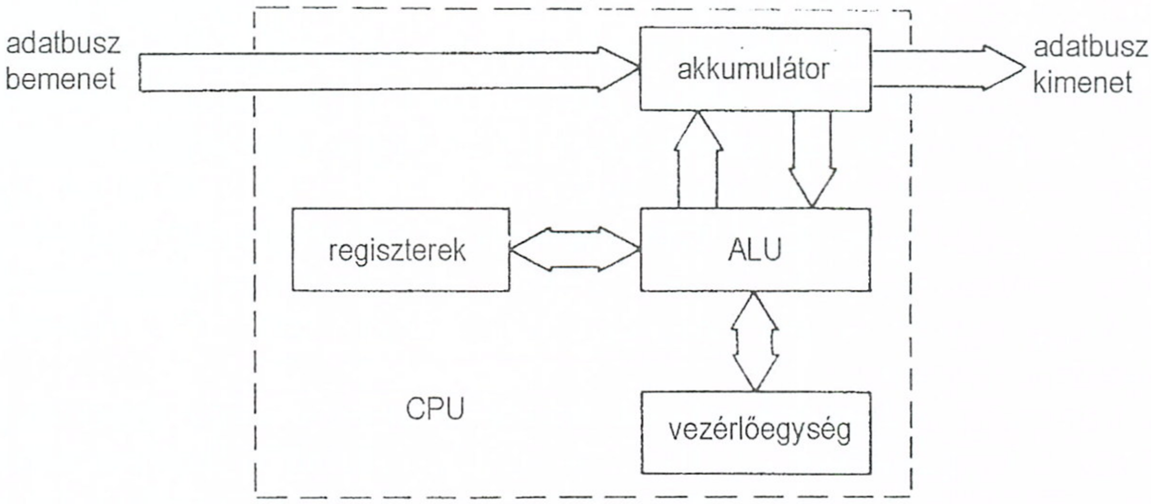
A regiszterek általában statikus memóriaelemek (például flip-flop áramkörök) valamilyen rendszeréből épülnek fel. (Ahol az összes regisztert statikus elemekből építik fel, az óraimpulzusok sorozatát leállíthatjuk anélkül, hogy a tárolt információkban veszteség keletkeznék.)

3.1.1. Tipikus regiszterek

Akkumulátor (Accumulator = A)

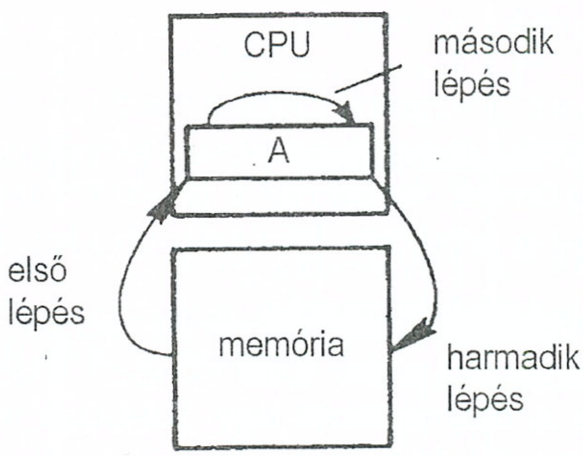
Az akkumulátor a számítógép megkülönböztetett számítási regisztere. Az aritmetikai és logikai műveletek operandusait, vagyis a műveletek tárgyát képező mennyiségeket, illetve ezeknek az eredményeit a CPU központi regiszterében, az akkumulátorában tároljuk.

A különböző műveleteket a CPU az akkumulátorban tárolt adatokon, vagyis az akkumulátor tartalmán végzi el (IV.1. ábra).



IV.1. sz. ábra: Adatszófolyam a mikroprocesszoron keresztül

A IV.2. ábrán látható, hogy az adatok kihozása a memóriából (olvasás) és az akkumulátorban történő elhelyezése, a művelet elvégzése és az eredmény bevitele (írás) a memóriába, három elemi lépést jelent.



IV.2. sz. ábra: A mikroprocesszor (CPU) akkumulátora és a memória között létrejövő adatforgalom lépései

Azt az időtartamot, ami alatt az adatokhoz hozzáférünk, vagyis ami a címkijelöléstől az olvasás, vagy írás műveletek befejezéséig eltelik, **hozzáférési**, illetve **elérési idő**nek nevezzük.

A jelenlegi számítógépekben a regiszterek hozzáférési ideje pár nanoszekundum. (1 nanoszekundum a másodperc ezer milliomod része (10^{-9} sec), ez alatt az idő alatt a fény mindössze 30 cm-t tesz meg.)

A CPU sebességét érdemes szemléletesen is összehasonlítani például az emberével. Ha a regiszter hozzáférési idejét 1 másodpercnek feleltetjük meg, akkor ehhez viszonyítva az az időtartam, amíg az ember leüt egy billentyűt a számítógép klaviatúráján kb. 30 év.

Megjegyzés: A legtöbb CPU-ban több akkumulátor, vagy akkumulátor típusú regiszter is van. Ekkor megfelelő szervezéssel elérhető, hogy egy memória hozzáférési idő alatt több CPU művelet is elvégezhető legyen.

A korszerű számítógépekben rendszerint egy vagy több regisztertömb is megtalálható 8, 16, ... max. 512 darab általános célú regiszterrel, melyet regisztertárnak nevezünk. E regiszterekben helyezhetők el ideiglenesen a műveletek operandusai és eredményei. Így csökkenthető a tárhozfordulások száma, ezáltal növelhető az utasításvégrehajtás sebessége.

Utasításregiszter (Instruction Register = IR),

Az utasításkódok kezeléséhez szüksége van a CPU-nak az utasításregiszterre, amelyben a program aktuálisan végrehajtott utasítását átmenetileg tárolja.

Utasításszámláló (Program Counter = PC)

Az utasításszámláló regiszter a soron következő utasítás címét tárolja. Az utasításszámláló tartalmát a program maga is módosíthatja. (Ez történik például a program által végrehajtott elágazási utasítások esetén.)

A program az utasításkódok címeinek sorrendjében hajtódik végre. Vagyis az utasításszámláló által címzett első memóriarekesz elérésekor kihozunk a memóriából egy utasítást, melynek következtében az utasításszámláló tartalma a kihozott utasítás hosszával megnő, és így a memóriának most azt a rekeszét címzi, ahol a program szerinti következő utasításkód található.

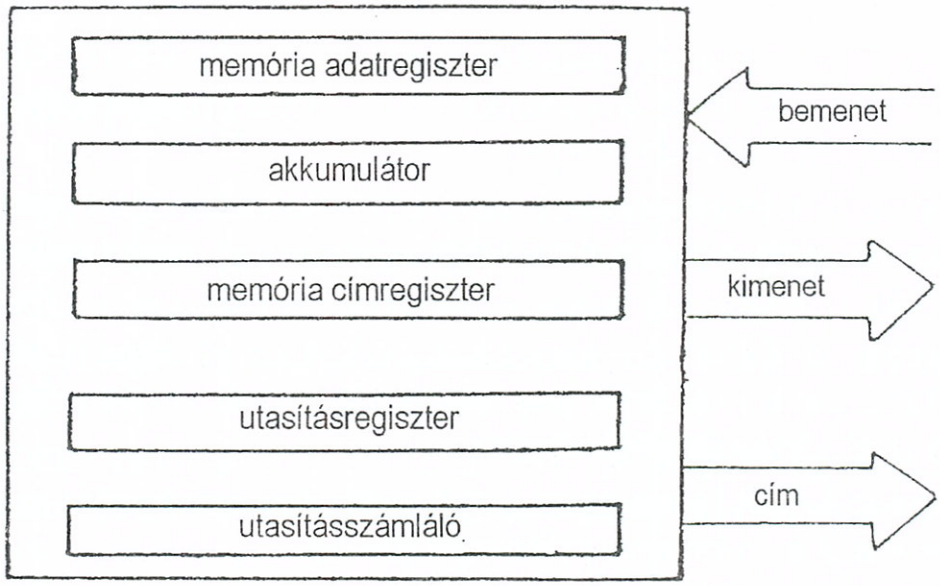
Memória címregiszter (Memory Adress Register = MAR)

Az adatok kiolvasásakor vagy beírásakor azonosított memóriarekesz címét a memória címregiszter tárolja. Ennek mérete függ a mikroprocesszor által címezhető memóriakapacitástól.

Memória adatregiszter (Memory Data Register = MDR)

A memória adatregiszter a memóriából kiolvasott vagy beírni kívánt adatok átmeneti tárolója, úgynevezett **pufferregiszter** (buffer

register). Általában a pufferregiszter feladata, hogy az adatok áramlási ütemének egyenetlenségeit, vagy az események időbeli előfordulásának különbségét kiegyenlítse abban az esetben, amikor az adatokat különböző sebességű hardver egységek között kell mozgatni. Az MDR a CPU és a főtár sebességének különbségét egyenlíti ki, mivel a főtár RAM memóriája kb. egy nagyságrenddel lassúbb a regiszterénél.



IV. 3. sz. ábra: Egy egyszerű mikroszámítógép tipikus regiszterei

A CPU-nak még két fontos regisztere van: az állapotregiszter és a veremmutató regiszter, ezeket a felhasználásukkal foglalkozó 3.4. és 4.1.4. fejezetekben fogjuk bemutatni.

3.1.2. A regiszterek használata, az utasításvégrehajtás elemi lépései

Hogyan használjuk a CPU regisztereit?

Annak érdekében, hogy a CPU regisztereinek használatát megértsük, még egyszer végigkövetjük az összeadás elemi lépéseit, lépésről-lépésre részletesebben bemutatva.

Figyeljük meg a regiszterek tartalmának változását az utasítások végrehajtásának menete során.

1. lépés: Az utasításszámlálóban található cím bekerül a memória címregiszterbe. Ez alapján kiolvassuk a memóriából az összeadó

utasítás gépi kódját, ez pufferelésre kerül a memória adatregiszterben. Innen a következő ütemben átkerül az utasításregiszterbe.

2. lépés: Az utasításszámláló regiszter tartalmát megnöveljük a kiolvasott utasítás hosszával (így ez most a következő végrehajtandó utasításra mutat).

3. lépés: A CPU értelmezi az utasításregiszterben található utasítás kódját, megállapítja, hogy az összeadás műveletét kell végrehajtani.

4. lépés: A CPU az utasításregiszterben található címek alapján meghatározza az első összeadandó címét a memóriában, és ezt beírja a memória-címregiszterbe.

5. lépés: A CPU kiolvassa az első összeadandó értéket a memóriából, ez bekerül a memória-adatregiszterbe, majd onnan az akkumulátorba.

6. lépés: A CPU az utasításregiszterben található címek alapján meghatározza a második összeadandó címét a memóriában, és ezt beírja a memória-címregiszterbe.

7. lépés: A CPU kiolvassa a második összeadandót a memóriából, és ez bekerül a memória-adatregiszterbe.

8. lépés: A CPU műveletvégző egységében (aritmetikai és logikai egységben – ALU) megtörténik az összeadás és ennek eredménye képződik az akkumulátorban

9. lépés: A művelet eredménye visszaírásra kerül a memóriába, az első összeadandó címének megfelelő helyre. (Az első összeadandó címe átkerül a memória címregiszterbe, az eredmény a memória-adatregiszterbe és innen kerül beírásra a memóriába.)

3.2. Aritmetikai logikai egység (ALU)

Az aritmetikai logikai egység a CPU-n belül a számítási és logikai műveleteket végzi el.

Az ALU alkalmas:

- bináris összeadásra,
- Boole-algebrai (logikai) műveletek végzésére,
- komplemensképzésre,
- bitsorozatok léptetésére (bitenként jobbra, balra).

Minden más aritmetikai és logikai művelet (például lebegőpontos számokkal végzett műveletek), amelynek elvégzése a CPU feladata, felbontható a fenti alapműveletekre.

Ennek megfelelően az ALU rendszerint fixpontos bináris összeadóból, komplementálóból, léptető regiszterből és a logikai műveleteket végző részből áll.

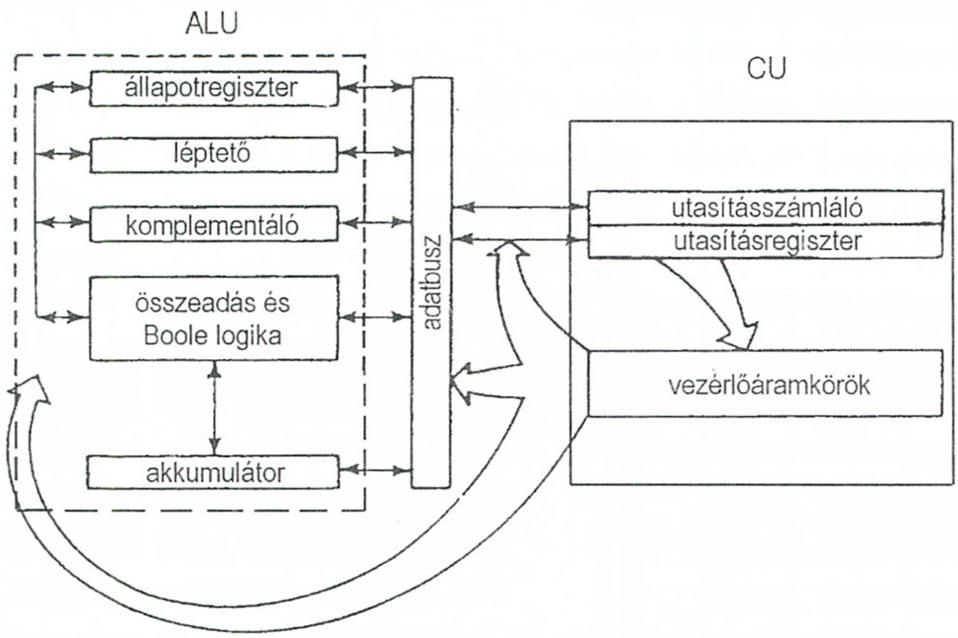
3.3. Vezérlőegység (CU)

A vezérlőegység funkciója bizonyos szempontból analóg a bábjátékos feladatával, aki úgy manipulálja a bábukat mozgató zsinegeket, hogy a bábu a megfelelő lépések szerint táncoljon. Egy mikroprocesszorban a regiszterek és az ALU a bábu, míg a vezérlőegység a bábjátékos.

A legtöbb mikroprocesszorchipben a vezérlőegység a teljes chipnek több mint a felét foglalja le, és rendkívül összetett funkcionális egység.

A vezérlőegység (Control Unit = CU) a CPU-n belül értelmezi az utasításokat és végrehajtásuk céljából összehangoltan vezérli a számítógép többi egységének működését úgy, hogy az események a programnak megfelelő helyes sorrendben és időben következzenek be. Biztosítja, hogy a megfelelő adatok, a megfelelő helyen és időben rendelkezésre álljanak. Indítja az áramkörök működését. Az utasításszámláló tartalma alapján a vezérlőegység kiolvastatja a memóriából annak a rekesznek a tartalmát, amely a soron következő utasítást tárolja. Ezt az utasítást értelmezi, azaz az utasítás műveleti kódrésze alapján meghatározza, hogy sorrendben milyen műveletet, milyen elemi lépésekben kell végrehajtani. Az utasítás címrésze alapján értelmezi, hogy milyen címen találhatók a műveletben résztvevő adatok, vezérli ezek kiolvasását, és a megfelelő regiszterbe történő továbbításukat. Az ALU-val végrehajtatja a megfelelő műveletet, biztosítja az eredmény megőrzését és beállítja az utasításszámláló új tartalmát.

A vezérlőegység és az aritmetikai logikai egység működésének lényegét a IV. 4. ábra mutatja be.



IV.4. sz. ábra: A mikroprocesszor funkcionális ábrázolása

3.4. Állapotinformáció

Az állapotinformációval a számítógép pillanatnyi állapotát jellemezzük. Ezeket az állapotokat egy regiszter bitjeinek 0-ra vagy 1-re állításával kódoljuk.

Jellegzetes állapotinformációk például a következők:

előjelbit:	értéke a numerikus adatokon végzett műveletek eredményének előjele szerint kerül beállításra. A bit értéke 1, ha a művelet eredménye negatív.
túlcsordulásbit:	a bit értéke 1, ha a numerikus adatokon végzett művelet eredménye túlcsordulás (azaz a művelet eredménye nem fér be a regiszterbe; pl. két szám szorzásánál)
átvitelbit:	értéke 1, ha van átvitel az előjelbit helyi értékére
nulla bit:	a bit értéke 1, ha a művelet eredménye 0
paritásbit:	paritáshiba esetén kerül beállításra 1-re

Az állapot-(státusz) információt az aritmetikai logikai egység hozza létre.

Az állapotbitek értéke általában **feltételes elágaztató utasításokkal** (lásd később) kérdezhető le és ettől függően változtatható meg az utasítás-végrehajtás sorrendje.

3.4.1. A jelzőbit

A jelzőbit (flag) olyan állapotinformáció, mely egy rendszer (hardverrendszer vagy program stb.) működési módjáról, működési feltételeiről vagy a működés közben előálló állapotokról (pl. eredmény=0, kiviteli/beviteli műveletek eredményes vagy eredménytelen végrehajtása stb.) ad felvilágosítást. Különbséget kell tenni hardver úton vagy szoftver úton (programváltozóként, illetve logikai változóként) megvalósított jelzőbitek között. A jelzőbitek beállíthatók, lekérdezhetők, illetve módosíthatók.

3.4.2. Állapotregiszter

Az állapotregiszter olyan regiszter, amely a központi egység, vagy a perifériaeszközök állapotával kapcsolatos információkat tartalmazza.

A CPU állapotregiszterének bitjei (status register) az ALU műveleti eredményei alapján kerülnek beállításra (előjelbit, túlcsordulásbit stb.).

4. A processzor utasításkészlete, címzési eljárások

A processzorok egyik fontos jellemzője – ami egy-egy feladatra való alkalmasságukat döntően meghatározza –, hogy hány és milyen típusú gépi utasítás végrehajtására képesek. A processzor számára értelmezhető utasítások összességét nevezzük **utasításkészlet**nek.

4.1. A gépi utasítás szerkezete

Az utasításkészletben különböző típusú gépi utasítások lehetnek, ezek felépítésükben is eltérhetnek. Ezt fejezi ki többek közt az utasítás szerkezete, mely meghatározza a processzor számára, hogy a gépi utasítás mely részét hogyan kell értelmezni.

Általában az utasítások felépítése a IV.5. sz. ábra szerinti.

műveleti rész	módosító rész	címrész

IV.5. sz. ábra: A gépi utasítás felépítése

A műveleti rész (operation code) az elvégzendő feladatot határozza meg a processzor számára. (Tehát például megadja, hogy két számot össze kell adni.)

A címrész (address field) a művelet végrehajtásához szükséges adatok helyét határozza meg a számítógép tárolójában.

A módosító rész esetenként a műveleti rész illetve a címzés értelmezéséhez ad kiegészítő információkat a processzor számára.

4.2. Címzési módszerek

A gépi utasítások abban is különböznek egymástól, hogy az adatokra való hivatkozást milyen formában tartalmazzák. Ezeket összefoglaló néven címzési eljárásoknak szokták nevezni.

Az utasítások címrésze általában nem a műveletben résztvevő adatok pontos rekeszsorszámát, az úgynevezett abszolút címet tartalmazza, hanem ezt a processzornak a címzési eljárásra utaló kiegészítő adatok alapján ki kell számítania.

Az elkövetkezendőkben a címzési eljárások közül áttekintjük

- az abszolút címzést,
- a relatív címzést,
- a közvetlen adat (literális) címzést,
- a veremcímzést,
- a közvetett címzést,
- az indexelt címzést.

4.2.1. Az abszolút címzés

Abszolút címzés esetében az utasítás címrésze, a műveletben részt vevő adatok, az operandusok valódi címét, azaz a memóriarekesz sorszámát tartalmazza, illetve a processzor egy regiszterére hivatkozik.

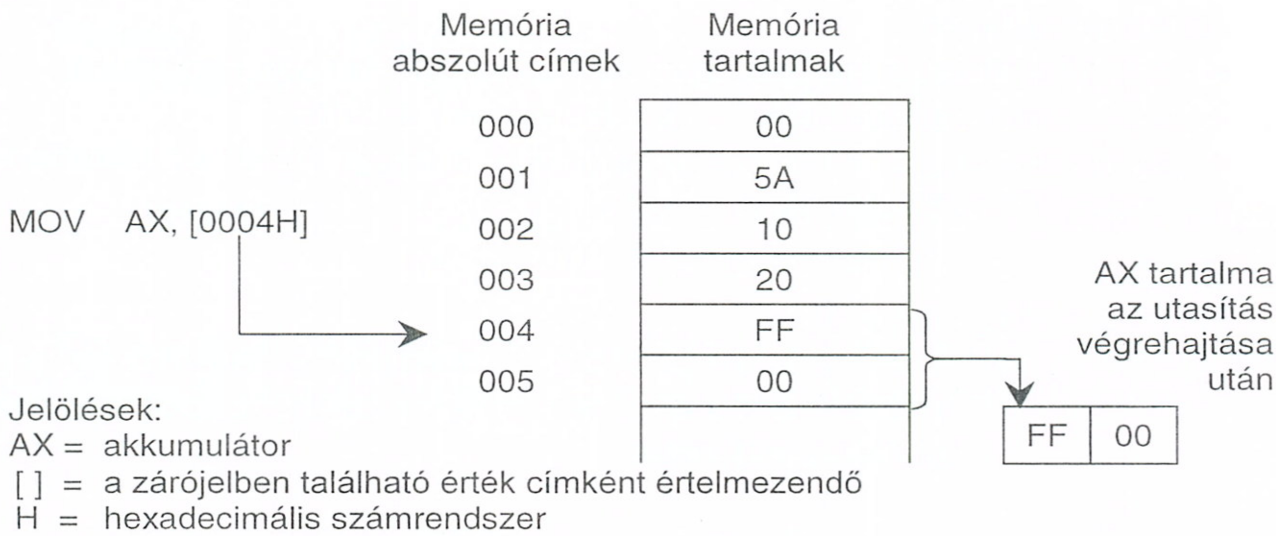
Példák: 1. MOV AX, DX

Jelentése: A DX regiszter tartalmát vidd át az akkumulátorba.

2. MOV AX, [0 0 4 H]

Jelentése: A tároló 004 hexadecimálisan értelmezendő (ezt jelenti a H) címéről két bájtot tölts be az akkumulátorba.

Az abszolút címzést mutatja be a IV.6. ábra, melynél a szerepeltetett számok hexadecimálisan jelölnek bináris értékeket.



IV.6. sz. ábra: Abszolút memóriacímzés

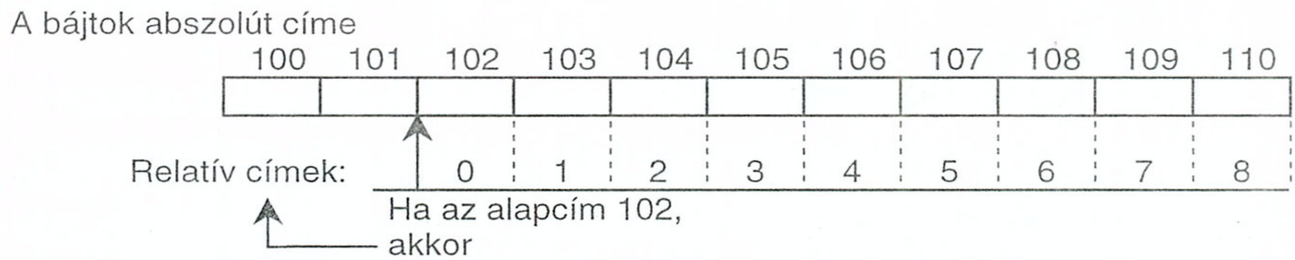
A regiszterek abszolút címzését a programokban gyakran alkalmazzuk, az abszolút memóriacímzést viszont csak nagyon ritkán.

Ennek az az oka, hogy a programokat nem mindig azonos memóriacímre töltjük be (hanem például onnan kezdődően, ahol van szabad hely a memóriában), így az egyes programbetöltési címtől függően a programutasításokban megcímzett adatok fizikai helye is megváltozik.

Az előző példánkban szereplő Intel Assembly utasítás 004H címe is lényegében relatív címzést takar (Szegmensen belüli relatív címzés).

4.2.2. Relatív címzés

Relatív címzés esetén a gépi utasítás-címrésze az adatnak valamilyen alapcímhez vagy báziscímhez viszonyított relatív címét tartalmazza. A tényleges fizikai memóriacímet ekkor a relatív cím és a báziscím összeadásával kapjuk meg.



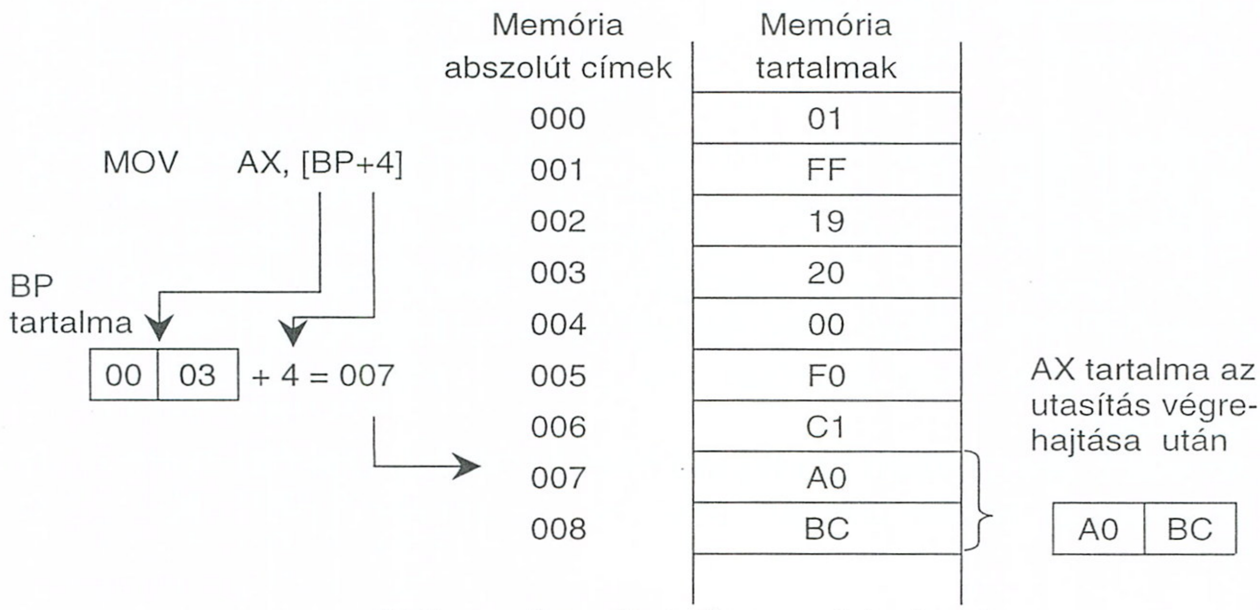
IV.7. sz. ábra: A relatív címzés elve

Az alapcím lehet
- egy kijelölt regiszterben, melyet bázisregiszternek nevezünk,
- a program kezdetének (töltési helyének) a címe
- az éppen végrehajtott utasítás (ekkor a bázis címet az utasításszámláló regiszter tartalmazza).

Példa

MOV AX, [BP + 4]

Jelentése: A BP bázisregiszterben található címértékhez adj hozzá 4-et és ettől a címtől kezdve 2 bájtot tölts be az akkumulátorba.



IV.8. sz. ábra: Relatív memóriacímzés

A relatív címzést alkalmazzuk akkor, ha biztosítani szeretnénk, hogy programjaink a memória tetszőleges helyére betöltve is futóképesek legyenek. Ezt elérhetjük azzal, hogy a program utasításaiban csak a program elejéhez (töltési címéhez) viszonyított relatív címeket alkalmazunk, és a program betöltését követően a BP bázisregiszterbe azonnal betöltjük a program elejének tárbeli címét.

4.2.3. Közvetlen adatcímzés

Ezt a címzési eljárást még közvetlen **értékadó címzés**nek, **álcímzés**nek és **literálcímzés**nek is szokták nevezni. A közvetlen adatcímzés lényege, hogy ebben az esetben az utasításban maga az az adat található meg, amellyel a műveletet végre kell hajtani.

Példák: 1. MOV AX, 0
Jelentése: Tölts be 0-t az akkumulátorba.

2. ADD AX, 010110B
Jelentése: Add hozzá az akkumulátorban található értékhez a 010110 bináris számot.

Ezt a címzési eljárást a programokban például a regiszterek konkrét számértékekkel konstansokkal való feltöltésére használhatjuk.

4.2.4. Veremcímzés (STACK-címzése)

Van egy olyan memóriacímzési mód, amelyet majdnem minden számítógép alkalmaz valamilyen formában, ez a veremcímzés.

A **verem** (stack), vagy egy regisztertár a CPU-ban, vagy egy kijelölt memóriaterület a főtárban, melyből az **utoljára beírt adatokat lehet először kiolvasni.**

Az adatforgalom a bemeneti sorrenddel ellenkező kimeneti sorrendű, így például az először beírt adatot lehet majd utoljára kiolvasni a veremből.

Az ilyen tulajdonságú memóriákat LIFO (LAST IN FIRST OUT = „utolsóként be, elsőként ki") szervezésűnek nevezzük. A verem egyik legfontosabb alkalmazása a CPU állapotának (az állapot és az utasításszámláló regiszter tartalma) elmentése abban az esetben, ha egy program végrehajtását átmenetileg fel kell függeszteni.

4.2.4.1. Memóriaverem

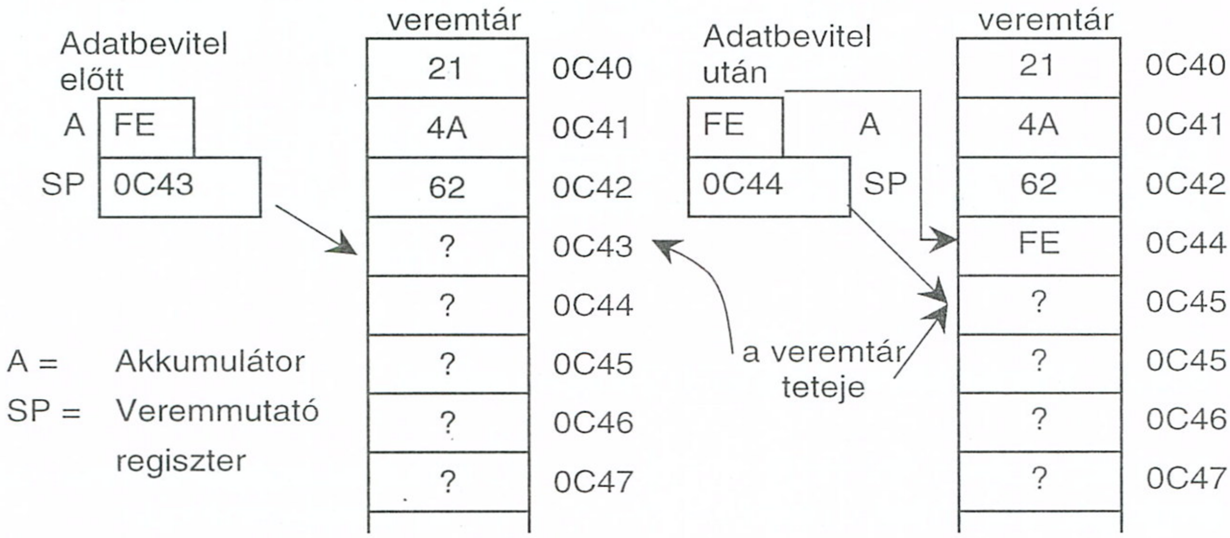
Az a verem legegyszerűbb megvalósítása, amikor a főtár egy részét tartalékolják veremműveletekre. A verem megcímzése ekkor egy regiszterrel történik, ezt nevezik **veremmutatónak** (STACK POINTER).

4.2.4.2. Veremtár műveletek

Két verem művelet van:

- **írás** a verem felső részébe (a tetejére), ezt nevezik adatbevitelnek, vagy PUSH-nak;
- **olvasás** a verem felső részéből, ezt nevezik „POP"-nak, vagy adatkihozatalnak.

Adatbevitel (PUSH)
A művelet a következő IV.9. sz. ábra segítségével jól követhető:



IV.9. sz. ábra: Adatbevitel – PUSH – művelet

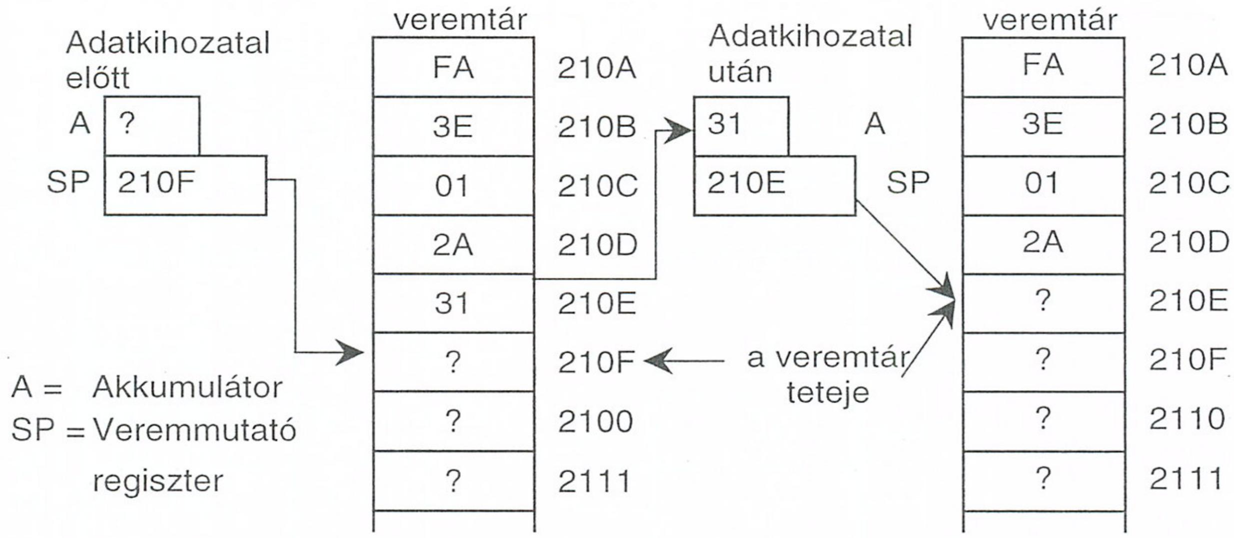
! Az adatbevitel művelete (írás a verembe) azt eredményezi, hogy az akkumulátorban (vagy más CPU regiszterben) lévő adatokat beírják abba a memóriarekeszbe, amelyet a veremmutató címez meg. Ezt követően a veremmutató tartalma automatikusan megnő (inkrementálódik), hogy megcímezze a következő üres tárolóterületet a verem felső részén.

Adatkihozatal (POP)

! Az adatkihozatal művelete ellentétes az adatbevitellel: először a veremmutató tartalmát eggyel csökkenti, hogy az a verembe utolsónak beírt adatot megcímezze, azután a veremmutató által megcímzett memóriarekesz tartalmát továbbítják az akkumulátorba vagy más CPU regiszterbe. Ezt a következő IV.10. sz. ábra mutatja be.

Figyeljük meg, hogy az adatkihozatal-művelet végén a veremmutató ismét megcímzi a verem felső részén lévő első szabad memóriatárolóhelyet, ugyanis azt tételezzük fel, hogy ha egy adatot kiolvasunk a verem felső részéből, akkor az adattárolóhely „kiürül”.

! A verem-memóriacímzés esetén **a veremmutató tartalmát növelni (inkrementálni) kell a beírás után és csökkenteni (dekrementálni) kell a kiolvasás előtt.**



IV.10. sz. ábra: Adatkihozatal – POP – művelet

Természetesen semmi akadálya nincs annak, hogy a műveletet ne a verem felső részén, hanem a verem alján végezzék el. Ebben az esetben azonban a beírás előtt **dekrementálni** és a kiolvasás után **inkrementálni** kell.

4.2.4.3. A kaszkád verem

Létezik egy másik veremszerkezet is. Ennél a megoldásnál a verem a CPU-ban van és korlátozott számú regiszterből áll.

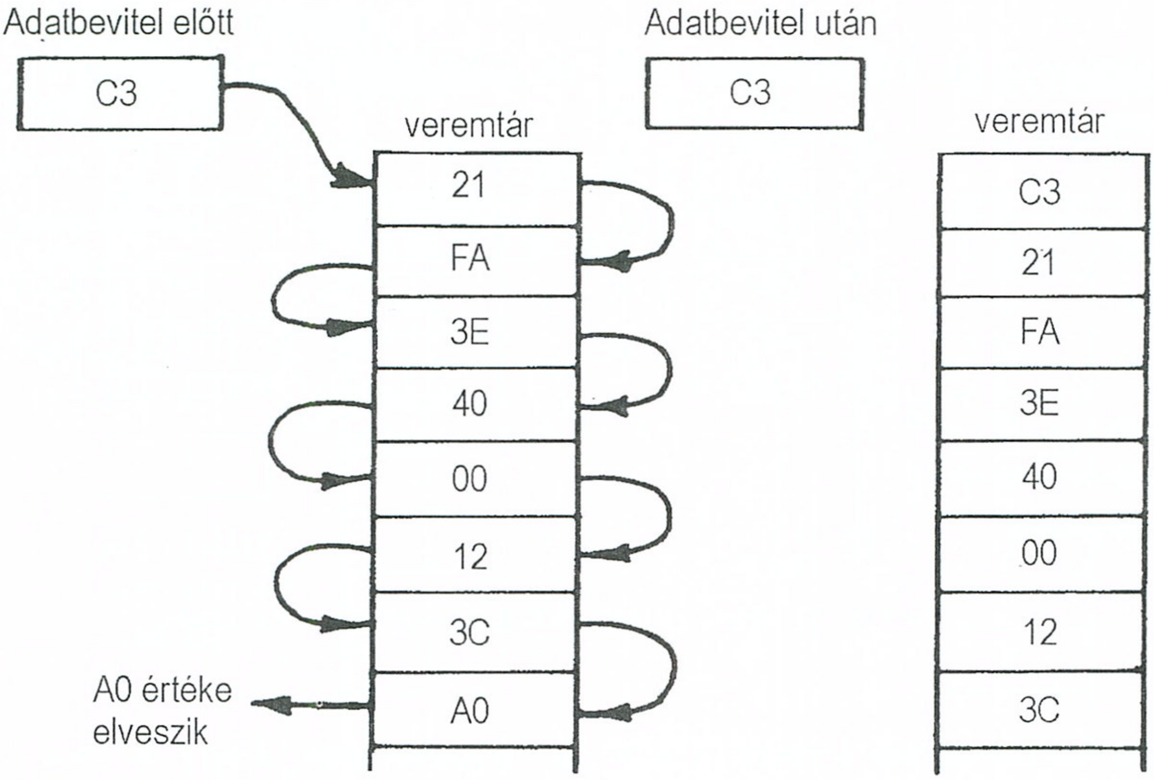
Adatbevitel (PUSH)

Ha egy adatbájtot beírunk a verembe, az a IV.11. ábrán látható **módon halad végig.**

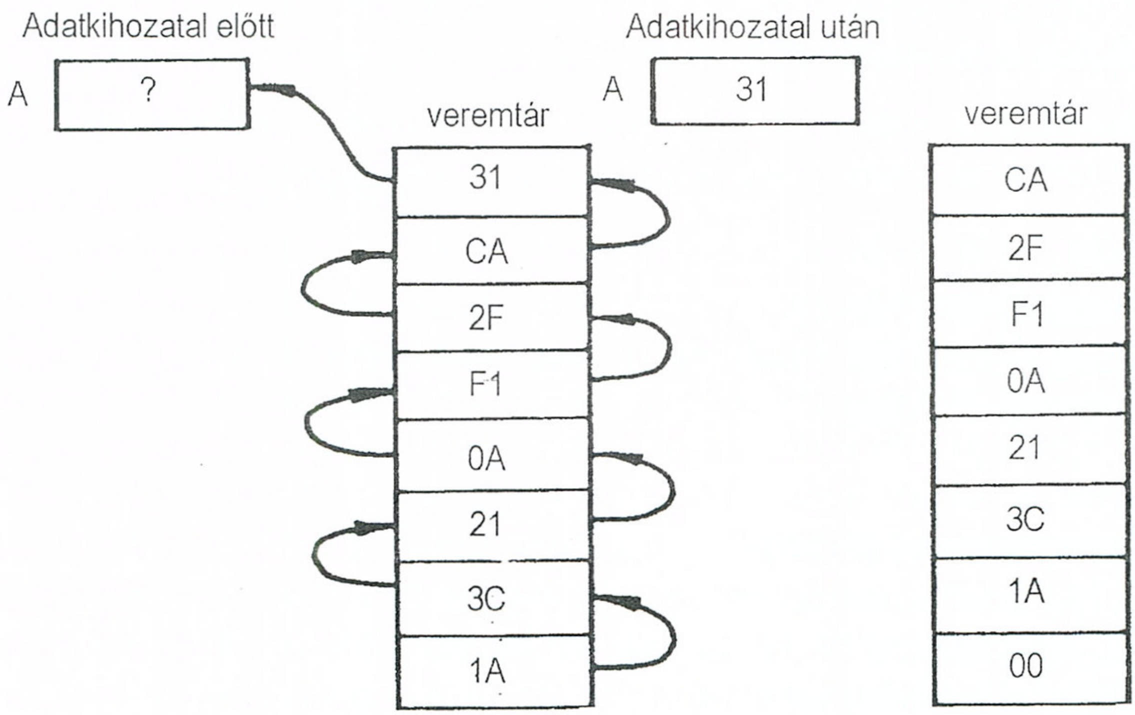
Adatkihozatal (POP)

Ha az adatkihozatal műveletét végezzük el a verem legfelső részéről, az adatok a IV.12. sz. ábrán látható módon haladnak végig.

Ehhez a veremszerkezethez nincs szükség veremmutatóra, mivel minden esetben az adatokat a verem legfelső részébe írják be, vagy onnan olvassák ki.



IV.11. sz. ábra: Adatbevitel a kaszkádba kapcsolt verembe



VII.12. sz. ábra: Adatkihozatal a kaszkádba kapcsolt veremből

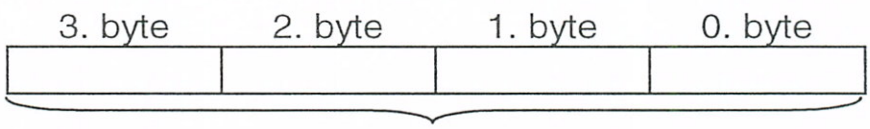
4.2.4.4. Mire használható a veremtár?

Szubrutinok

A legtöbb programban vannak olyan gyakran alkalmazott utasítássorozatok, amelyeket többször felhasználunk a program végrehajtása során, de csak egyszer tárolunk a programmemóriában. Ezeket a programrészeket (rutinokat) **szubrutin**nak nevezzük.

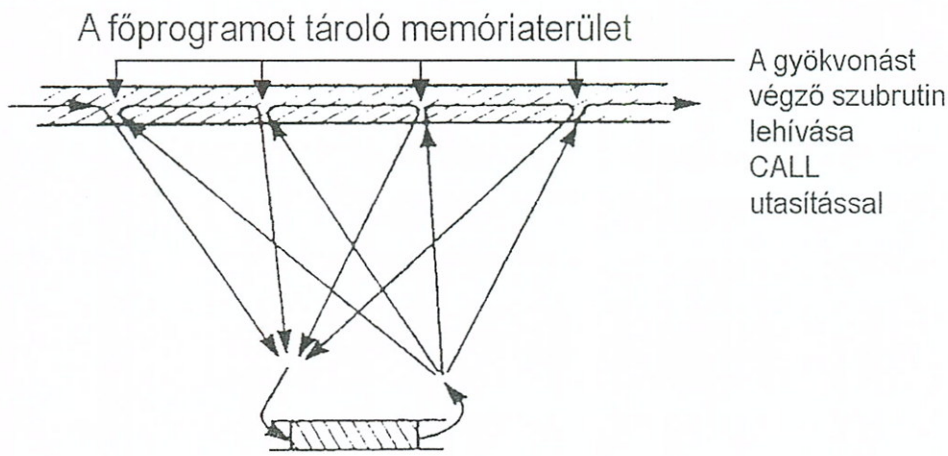
Nézzünk egy példát

Tegyük fel, egy programban többször előfordul, hogy olyan 32 bites számokból, amelyek négy egymással összefüggő bájtot foglalnak el, négyzetgyököt kell vonnunk.



IV.13. sz. ábra: Egy 32 bites szám

Az ilyen típusú aritmetikai művelet leghatékonyabban úgy végezhető el, ha külön programot írunk a gyökvonás elvégzésére. Minden esetben, amikor a programban el akarjuk végezni a gyökvonást, egy olyan utasítást alkalmazunk, amely lehívja (CALL) a gyökvonást végrehajtó szubrutint. A szubrutin lehívás a IV.14. sz. ábrán látható módon ábrázolható.



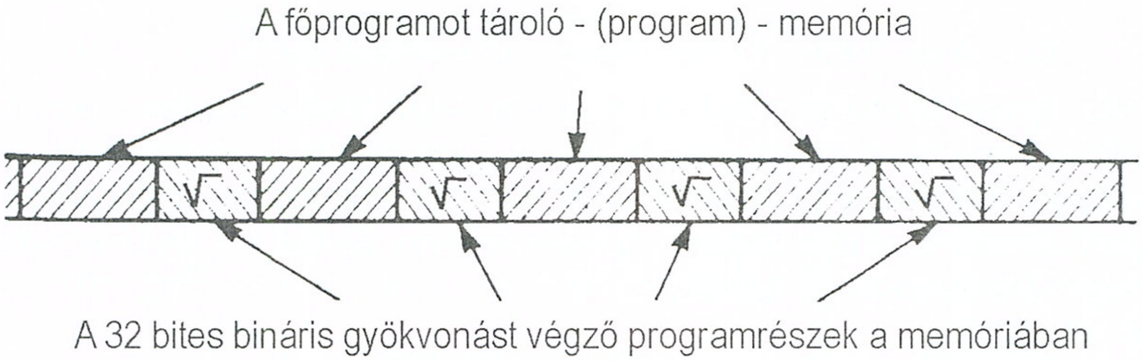
A gyökvonást végző szubrutin (program) memóriaterülete
(a nyíl jelöli az utasítás-végrehajtás sorrendjét)

IV.14. sz. ábra: Szubrutin lehívás – CALL – menete

Látható, hogy a szubrutinnak mindig át kell adni a 32 bites szám kezdőcímét, valamint egy másik tárolóhely címét, ahova a szubrutin a kiszámított négyzetgyök értékét elhelyezheti. Ezeket a **szubrutin paraméter**einek nevezzük.

Mivel – mint ezt a IV.14. sz. ábrán is láthatjuk – a főprogramot mindig a CALL utasítást követő utasítással akarjuk folytatni, a paraméter átadás, átvétel mellett meg kell jegyezni a főprogramba való visszatérés címét is.

Fontos felhívni arra a figyelmet, hogy ha nem alkalmazunk szubrutinokat, akkor a gyökvonás elvégzésére szolgáló utasítássorozatot minden alkalommal meg kell ismételni a programban (tehát például a programozónak annyiszor ismételten le kell írni), amikor a főprogramban 32 bites gyökvonást akarunk elvégezni. Így a program szubrutin nélkül a következő lenne:

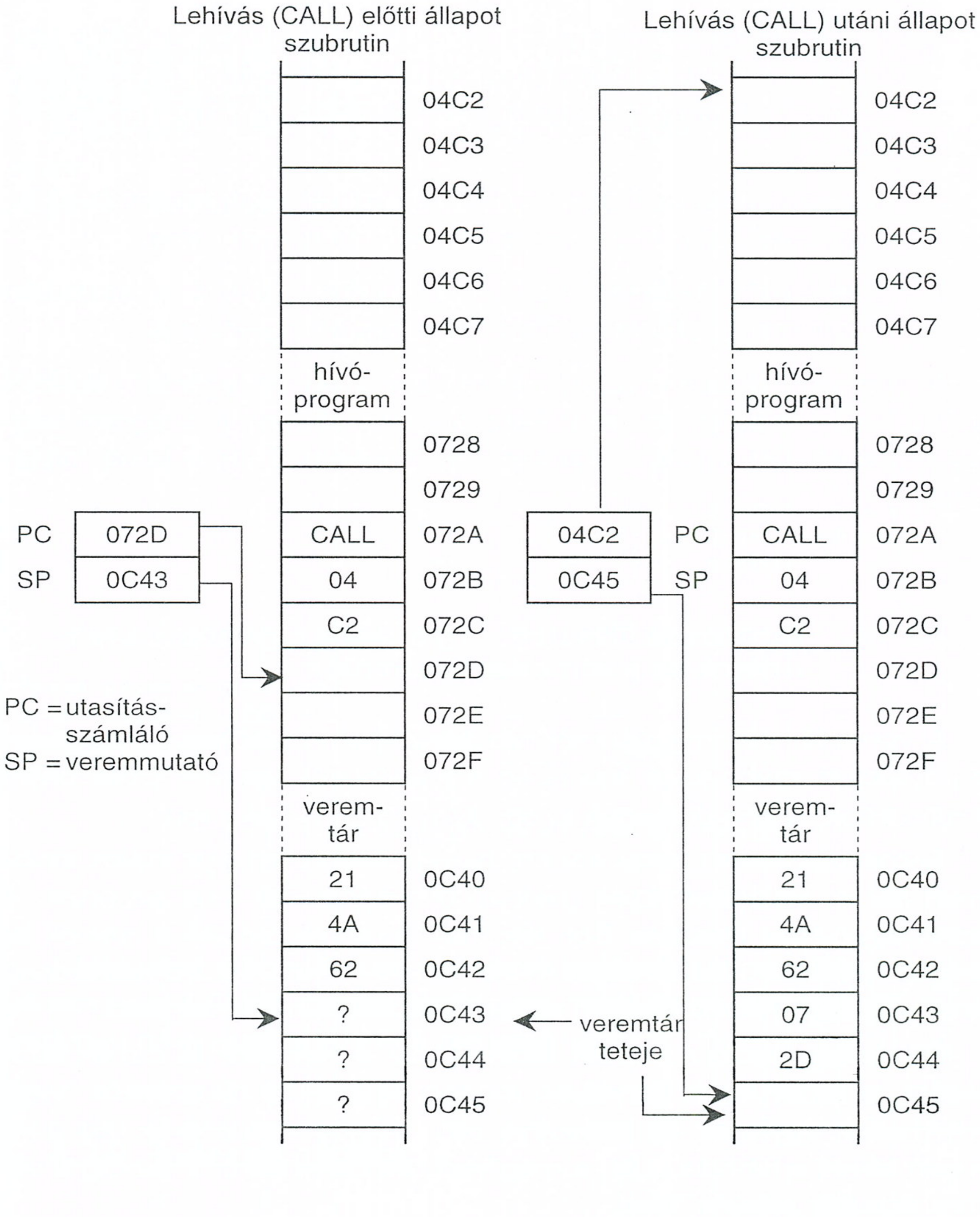


IV.15. sz. ábra: 32 bites gyökvonás a főprogramban szubrutin nélkül

Látható, hogy ez a változat a gyökvonást végző utasítások többszöri leírása mellett több memóriát is elfoglalna.

A szubrutin visszatérési címeket (azaz melyik címen kell a főprogramot folytatni a szubrutin befejezése után) legtöbbször veremben tároljuk.

A gyökvonás szubrutinjának lehívása ekkor a következő módon hajtható végre (IV.16. ábra). (Az ábra példájában a memóriarekeszek 8 bitesek, a címek mind 16 bit hosszúak, így két-két bájt szükséges minden egyes cím tárolására. Ezért a CALL utasítás 3 bájt helyet foglal el.)



IV.16. sz. ábra: 32 bites gyökvonás szubrutinjának lehívása veremhasználattal

A CALL szubrutinhíváskor

- az utasításszámláló regiszter (PC) tartalma (mely a CALL utasítás utáni 072D címre mutat) a SP veremmutató regiszterben lévő 0C43 címre (ez a verem aktuális teteje) 2 bájton elmentésre kerül,
- a veremmutató-regiszterben lévő címet két bájttal megnöveljük, az új 0C45 érték lesz a verem aktuális tetejének címe,
- azután az utasításszámláló regiszterbe bekerül a szubrutin 04C2 címe, ahol folytatódik az utasítások végrehajtása.

A szubrutinból való visszatéréshez csak a verem legfelső részéből kell a felső két bájtot az utasításszámlálóba átvinni. A program végrehajtása ezután a CALL utasítás után következő utasítással folytatódik.

4.2.4.5. Egymásba illesztett – egymásba ágyazott – szubrutinok és a veremtár használata

A veremtár alkalmazásának hasznosságát különösen az egymásba ágyazott szubrutinok esetében érzékelhetjük igazán. Ezek olyan szubrutinok, amelyek maguk is egy szubrutint hívnak meg.

Semmi szokatlan nincs abban, ha egy szubrutint egy másik szubrutin hív le. Valójában igen gyakran öt vagy annál több szubrutin is lehet egymásba illesztve. Sőt, léteznek olyan matematikai rutinok is, amikor a szubrutin önmagát hívja le. Az olyan típusú szubrutint, amely le tudja hívni önmagát, **rekurzív szubrutin**nak nevezik.

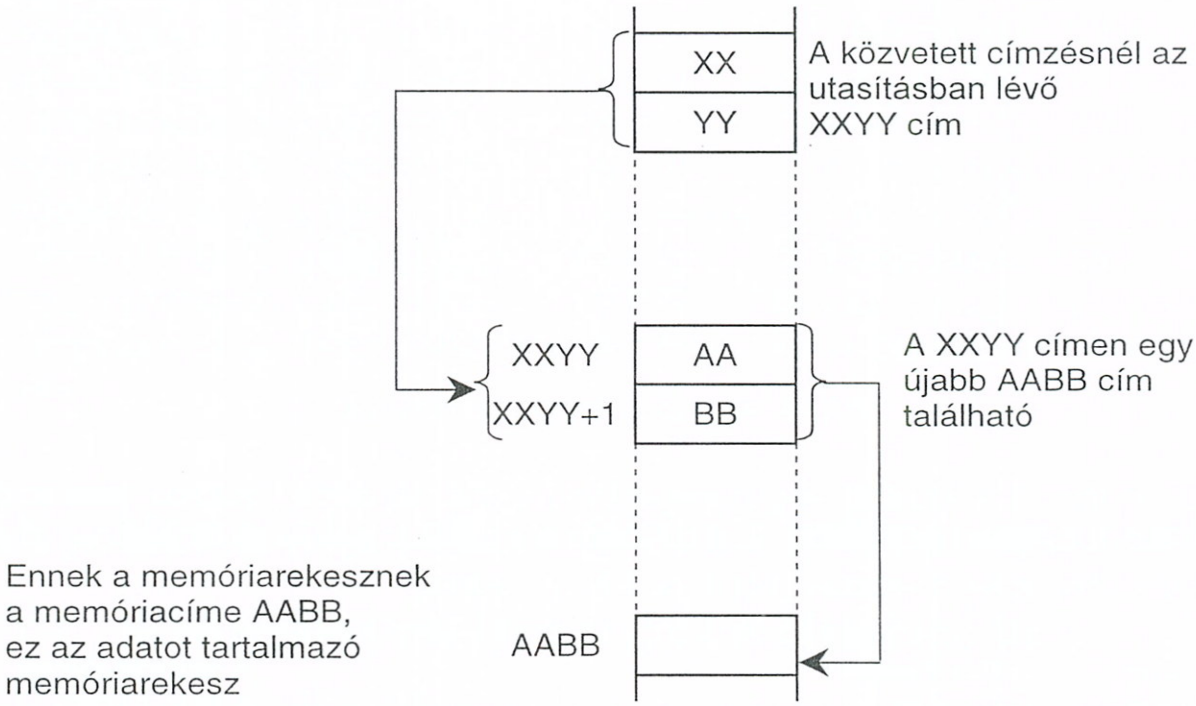
Mindaddig, amíg a vermet alkalmazzák a visszatérési címek tárolására, a szubrutinok bármilyen módon egymásba illeszthetők, vagy lehívhatják egymást: ha biztosítva van, hogy a visszatérés útja pontosan követi a lehívás útját, a megfelelő visszatérési cím mindig a verem aktuális „tetején” lesz.

A mikroszámítógép működésének és programozásának megértéséhez tényként kell elfogadni, hogy a verem biztosítja azt, hogy a szubrutinból való visszatérés útja pontosan fordítottja legyen a szubrutin lehívások útjának.

4.2.5 Közvetett címzés

A közvetett cím egy olyan memóriaterületet címez meg, amely az adat tényleges címét tartalmazza (IV.17. ábra).

A közvetett címet sokszor **pointer**nek (**mutató**nak) is szokás nevezni.

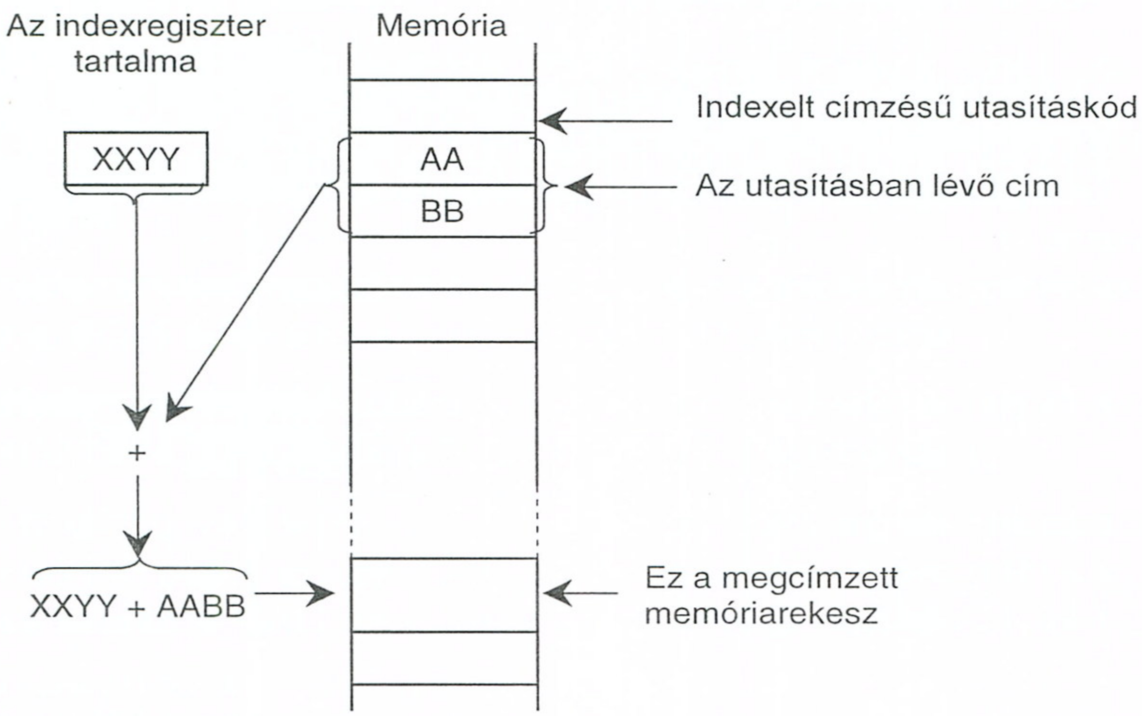


IV.17. sz. ábra: Közvetett memória címzés

A közvetett címzéshez a gépi utasításokban gyakran regisztereket alkalmazunk. Ekkor az utasításban egy regiszterre hivatkozunk, melyben a keresett adat címe megtalálható.

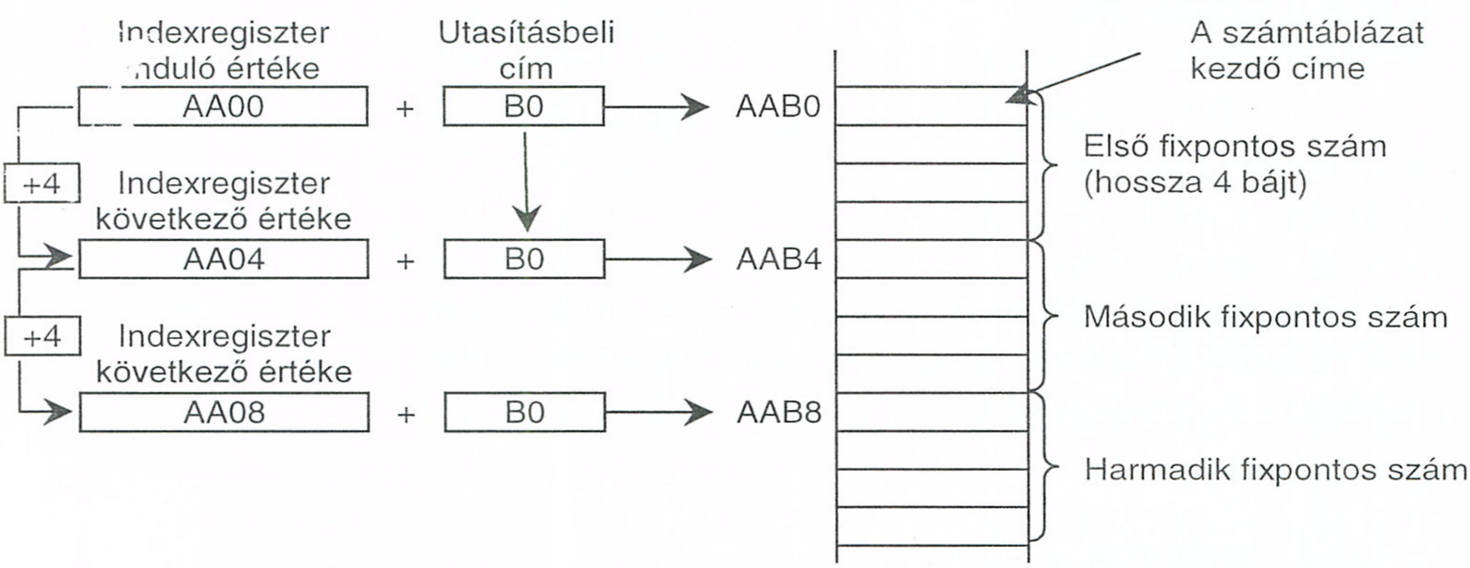
4.2.6. Indexelt címzés

Indexelt címzésnél a tényleges címet úgy kapjuk meg, hogy az utasításban lévő címhez (ez például egy vektortáblázat kezdőcíme a memóriában) hozzáadjuk az indexregiszterben lévő értéket. Ez a következő módon ábrázolható:



IV.18. sz. ábra: Indexelt címzés

! Az indexelt címzést leggyakrabban több azonos típusú elemből (például 4 bájtos fixpontos számból) álló táblázatok (ún. tömbök) feldolgozására alkalmazzuk (lásd IV.19. sz. ábra).



IV.19. sz. ábra: Számtáblázat címzése indexeléssel

Látható az ábrán, hogy minden egyes lépésben 4 bájttal, a táblázatban lévő fixpontos számok hosszával megnöveljük az indexregiszter értékét, így az indexelt címzéssel a táblázatban lévő összes szám címét lépésről-lépésre előállítjuk.

B. A MIKROPROCESSZOR ALAPÚ RENDSZER

Az eddigiek során röviden ismertettük a mikroprocesszorok fontosabb jellemzőit. Most megvizsgáljuk, hogy a mikroprocesszor család elemeiből hogyan történik a rendszerépítés. Mint már említettük, a mikroprocesszor önmagában működésképtelen, ahhoz, hogy céljainknak megfelelően használható digitális rendszerünk legyen, a mikroprocesszorainkat ki kell egészíteni memóriákkal, bemeneti-kimeneti egységekkel, más elemekkel. Az ily módon felépített rendszert mikroszámítógépnek nevezzük. A gyártó cégek az egyes kiegészítő funkcionális egységeket, amelyből a különböző igényeknek megfelelő mikroszámítógépek felépíthetőek, készen, tokozva forgalmazzák. Kialakult a családelv. Egy-egy család – ami mikroprocesszorból, memóriákból, bemeneti- kimeneti (I/O) egységekből, óragenerátorból, vezérlőkből stb. állhat – különféle mikroszámítógép konfiguráció megvalósítását teszi lehetővé. A családtagok egymással kompatibilisek.

1. Memóriák csatlakoztatása a mikroprocesszorhoz

1.1. ROM (Read Only Memory) csak olvasható tár

A ROM üzemszerű használatkor csak olvasható tároló, tartalmát általában a gyártó írja be. Leggyakrabban általános vezérlési funkciókat megvalósító programok tárolójaként alkalmazzák. Egyik megvalósítási formája, pl. planáreljárás során megfelelő maszk segítségével programozott, MOS tranzisztorokból álló tár.

A ROM memóriákat nagyon egyszerűen lehet a központi egységhez (CPU) illeszteni. Ha megvizsgálunk egy ROM tokot, akkor általában a IV.20. ábrán feltüntetett alábbi csatlakozópontokat találjuk rajta:

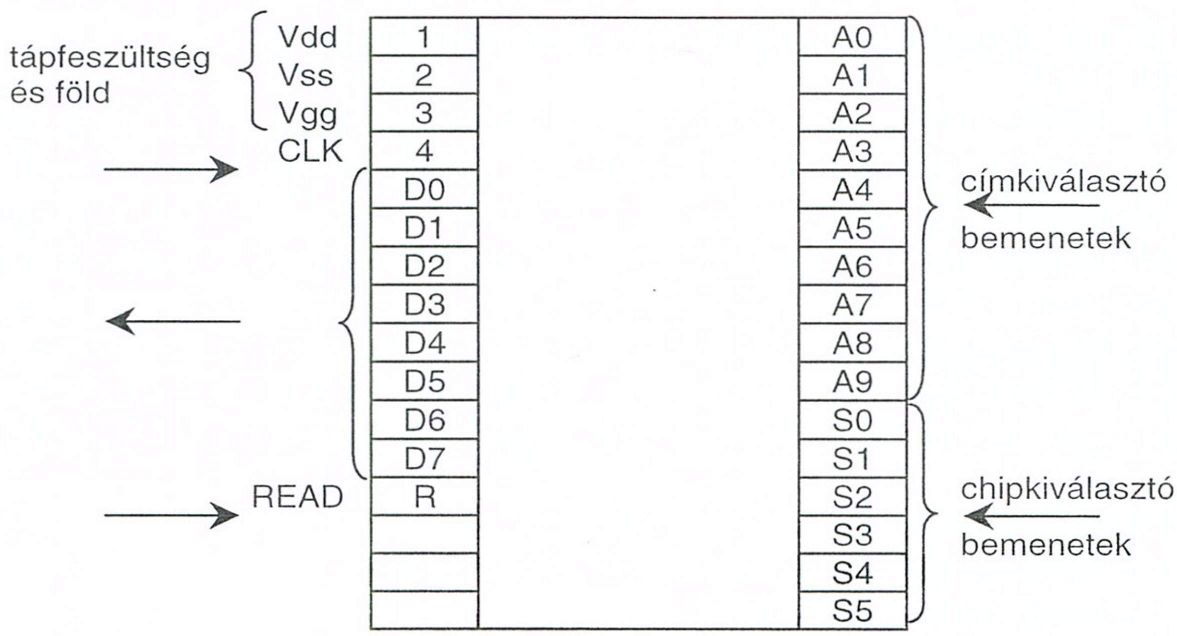
- címkiválasztó bemenetek (A0-A9);
- memóriachip-kiválasztó bemenetek (S0-S5);
- adatkimenetek (D0-D7);
- READ (olvasás) jelbemenet (R);
- órajelbemenet (CLK);
- tápfeszültség- és föld-csatlakozók (Vdd, Vss, Vgg)

A cím- és memóriachipet kiválasztó bemeneteken keresztül tudjuk elérni (kiolvasni) az éppen igényelt információt, vagyis egy rekesz

tartalmát. A megcímzett memóriarekesz tartalma a sínrendszer adatvezetékein keresztül kerül átvitelre a központi egységbe (CPU). Maga a kiolvasási művelet a READ (olvas) vezérlőjel hatására megy végbe. A műveletvégzés alatt a CPU és a ROM működése szinkronban van, a szinkronizálást a CLK órajel végzi.

Ha a mikroszámítógépünk csak egy mikroprocesszorból (CPU) és egy ROM-ból áll, a két eszköz nagyon egyszerűen – közvetlenül – összekapcsolható egymással. Általában azonban több eszköz szükséges ahhoz, hogy az igényeinknek megfelelő mikroszámítógéprendszerünket megalkossuk, amihez a szükséges jelkapcsolatokat egy, az eszközöket összekötő sínrendszeren – szokásos elnevezése szerint buszrendszeren – keresztül lehet célszerűen létrehozni. A továbbiak során az egyszerűség kedvéért 16 bites címzésű és 8 bites memóriarekeszeket tartalmazó számítógépet fogunk feltételezni (ez 16 bites címsínt és 8 bites adatsínt feltételez).

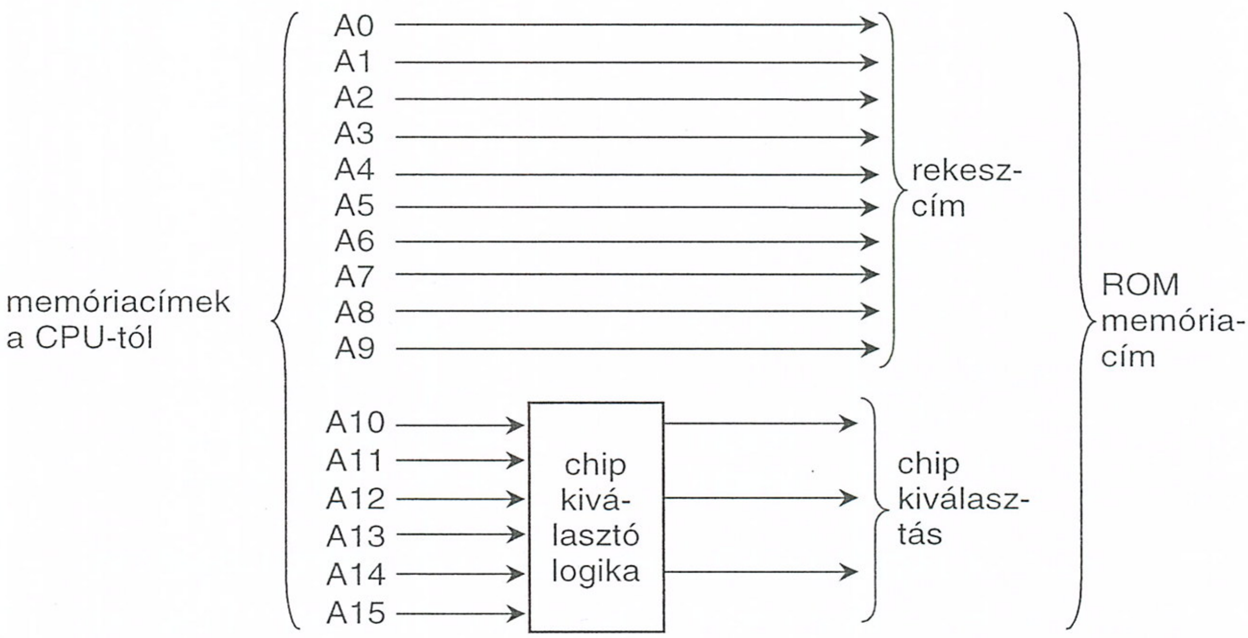
A mai számítógépek buszrendszere természetesen ennél jóval több adat egyidejű továbbítását is lehetővé teszi. Például a Pentium esetében a sín címvonalainak száma 32, az adatsín vonalainak száma pedig 64. Ez 64 bit-ből álló adatok egyidejű továbbítását biztosítja, és ennek megfelelően a memóriachipek csatlakozópontjainak száma is 32 bit cím- és 64 bit adatkimenetre módosul.



IV.20. sz. ábra: A ROM (fixmemória) csatlakozó pontjai

A CPU 16 címző jele közül 10 jel a rekesz címzésére, 6 jel pedig annak a ROM chipnek a kiválasztására szolgál, amelyben a rekesz

tárolva van. A címvezetékeknek ebből a felosztásából azonnal következik, hogy a ROM chip csak 1024 rekesz kapacitású lehet. Ha a CPU nagyobb helyi értékű 6 címvezetékének (A10-A15) jelei megegyeznek a ROM eszközünk azonosító kódjával (S0-S5), akkor a tárolót kezelő logika a kisebb helyi értékű címvezetékeken (A0-A9) található jelek alapján megjeleníti az adatvezetékeken (D0-D7) az 1024 közül kiválasztott egy memóriarekesznek a tartalmát.



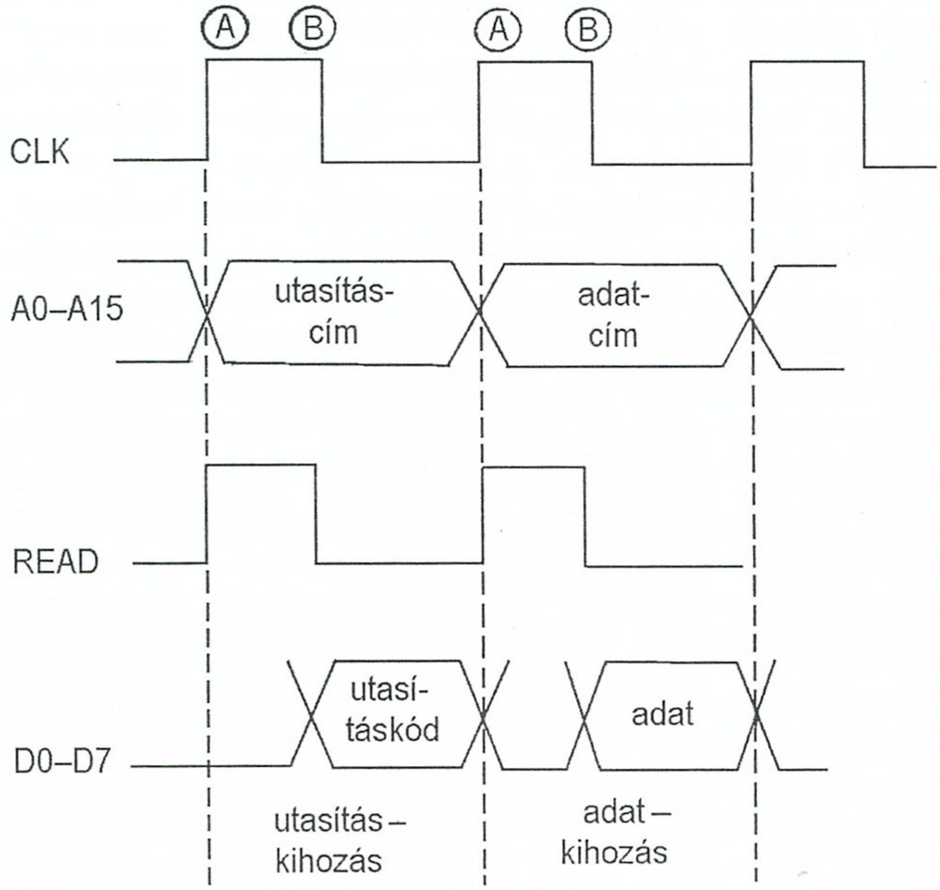
IV.21. sz. ábra: ROM- és/vagy RAM-kiválasztó logika

Vizsgáljuk meg egy ROM-CPU rendszerben a memória olvasási – READ – utasítás időzítését.

A vizsgált pontokat A és B jellel jelöltük (IV.22. ábra).

Minden művelet egy logikai 1-es bemenő CLK órajellel kezdődik („A" pont) és amikor a CPU kimenetein egy cím megjelenik, ugyanabban az időpontban a READ vezérlő jel logikai 1-re vált. Ezeket a jeleket a ROM a busz-(sín) rendszeren keresztül (IV.22. ábra) fogadja.

Ha a nagyobb helyi értékű 6 címvezeték – A10-A15 – jelei megegyeznek a ROM kiválasztó – S0-S5 – kóddal, akkor a ROM chip logikája az A0-A9 által címzett memóriarekesz tartalmát – ami utasítás vagy adat lehet – abban az időpontban helyezi rá a D0-D7 adatbuszra (adatsínre), amikor a IV.22. ábra szerint a „B" pontba érünk. Ekkor a CLK órajel és a READ jel logikai 0-ra vált. Az adat addig marad a D0-D7 adatbuszon, amíg a CLK órajel újra logikai 1 nem lesz.



IV.22. sz. ábra: A READ (memóriaolvasás) utasítás időzítése

1.2. RAM (Random Access Memory) közvetlen elérésű írható-olvasható tár

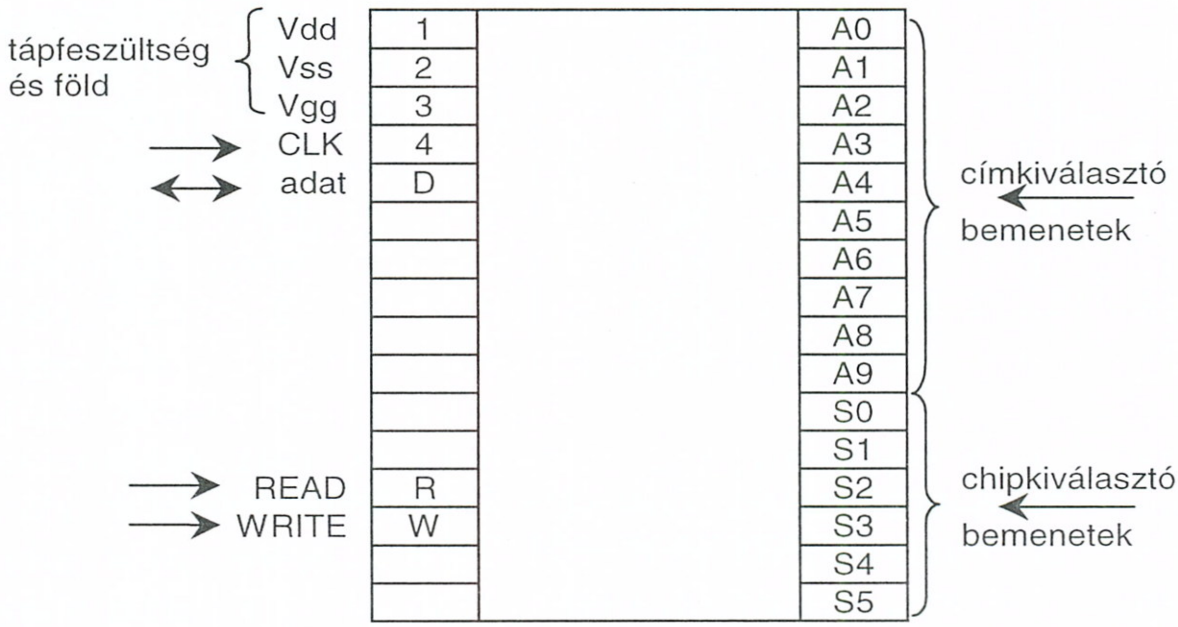
! A RAM-ba üzemszerűen beírható és belőle üzemszerűen kiolvasható az adat, ezért a RAM-ot kezelő illesztő logika sokkal bonyolultabb, mint ahogy azt a ROM memóriáknál láttuk.

A RAM-nak működéséből adódóan képesnek kell lennie arra, hogy az adatokat felvegye a külső vezetékekről, és elhelyezze azokat a címzett memóriarekeszbe, valamint arra is, hogy a kívülről címzett memóriarekesz tartalmát kiolvassa és elhelyezze a buszrendszer adatvezetékein.

Egy RAM chip tokján általában a IV.23. ábrán bemutatott alábbi csatlakozó pontokat találjuk:

- címbemenetek (A0-A9);
- memóriachip-kiválasztó bemenetek (S0-S5);

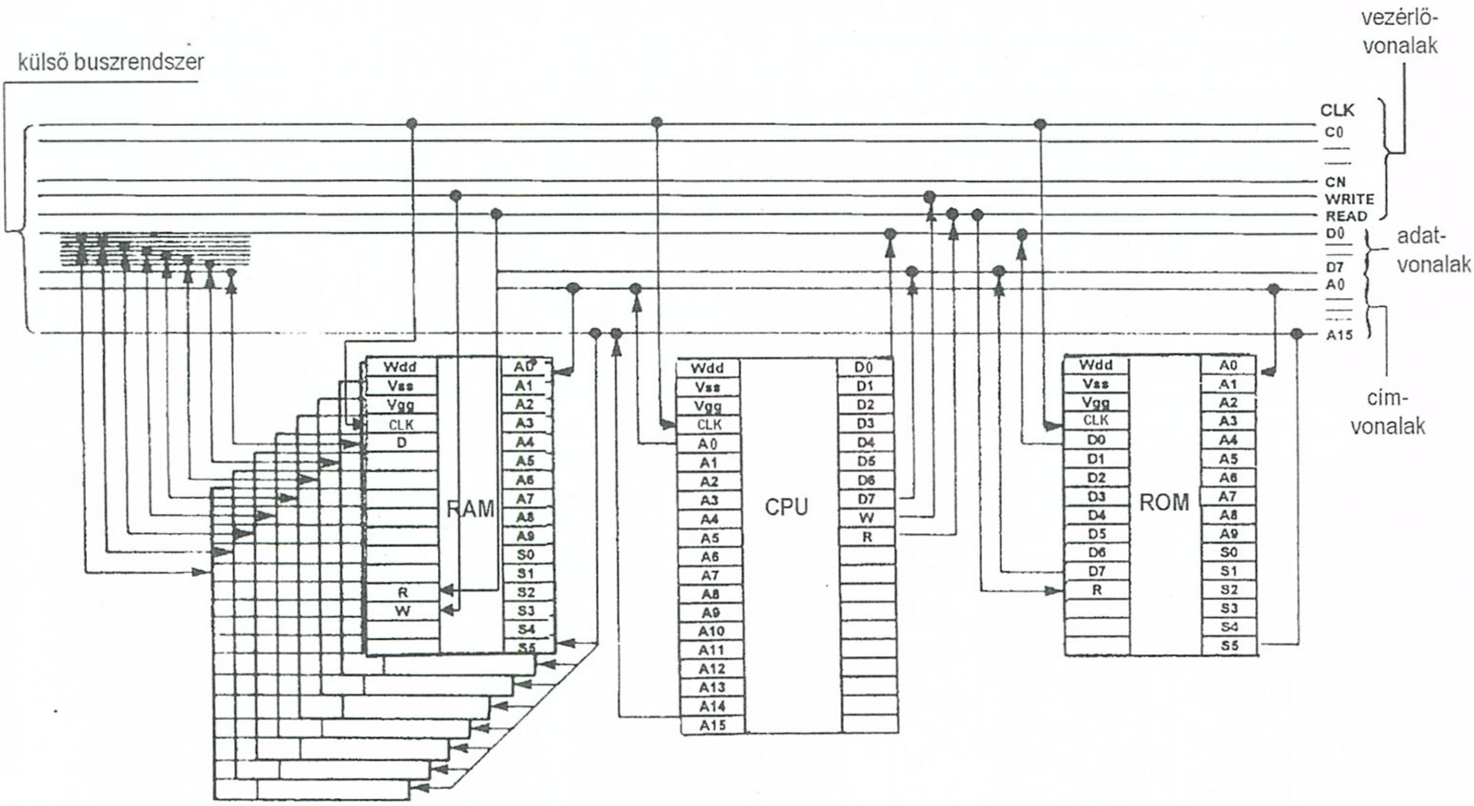
- adat-bemenetek/kimenetek (D);
- READ (olvasás) jelbemenet (R);
- WRITE (írás) jelbemenet (W);
- órajelbemenet (CLK);
- tápfeszültség- és föld-csatlakozók (Vdd, Vss, Vgg).



IV.23. sz. ábra: RAM (írható/olvasható memória) csatlakozó pontjai

A csatlakozások funkciói jórészt megegyeznek a ROM-nál ismertetettekkel. Eltérés az írási művelet elvégzéséből adódik, ami miatt az adatbusz (D) kétirányú adatátvitelt tesz lehetővé a CPU és a RAM között. A beírási művelet a WRITE (ír) vezérlőjel hatására megy végbe.

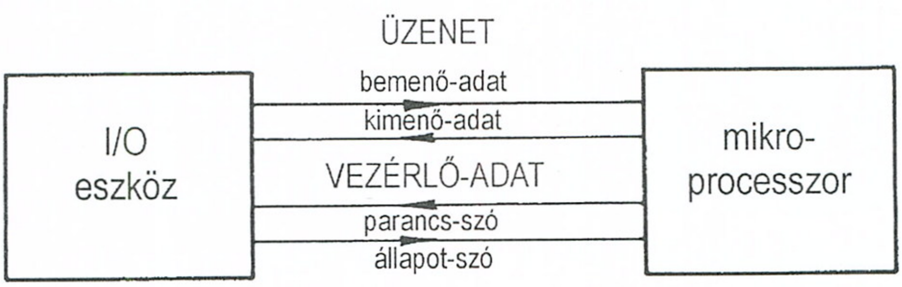
Egy rekeszhez tartozó bitek tárolására általában több RAM chipet használnak. A IV.24. ábrán bemutatott konfigurációban (8 bites rekeszhosszúság) 8 db RAM chip van. Egy RAM chipnél egy adatcsatlakozót (D) használnak fel, a RAM chipek csak egy-egy adatvezetékkel csatlakoznak az adatbuszra (D0-D7 adatvonalakra). Ezért a chipkiválasztó (beazonosító) kódjuk megegyezik, így a 8 db chipből kialakított RAM memória egy összefüggő egységet, memóriamodult képez. Tehát a 8-bites rekesz adatainak tárolását 8 chip végzi, mindegyik 1 bit tárolásáért felelős.



IV.24. sz. ábra: CPU, RAM és ROM csatlakoztatás a buszrendszeren keresztül

2. Adatátvitel a mikroszámítógép és a hozzá csatlakoztatott perifériális egységek között

Az input/output (I/O) eszközök és a mikroprocesszor között a sínrendszeren keresztül különböző adatok és vezérlőjelek áramlanak. Ezt szemlélteti a IV.25. ábra.



IV.25. sz. ábra: Adatfolyam a mikroprocesszor és a bemeneti/kimeneti (I/O) eszközök között

2.1. Interfész, I/O PORT

A mikroszámítógép központi része és a hozzá csatlakozó I/O egységek közötti adatátvitel céljából biztosítani kell az egyes egységek illesztését és a rendszer összehangolt működését. Ezért a mikroszámítógép a bemeneti és a kimeneti készülékekhez vezérlő és illesztő – interfész – áramkörökön keresztül csatlakozik.

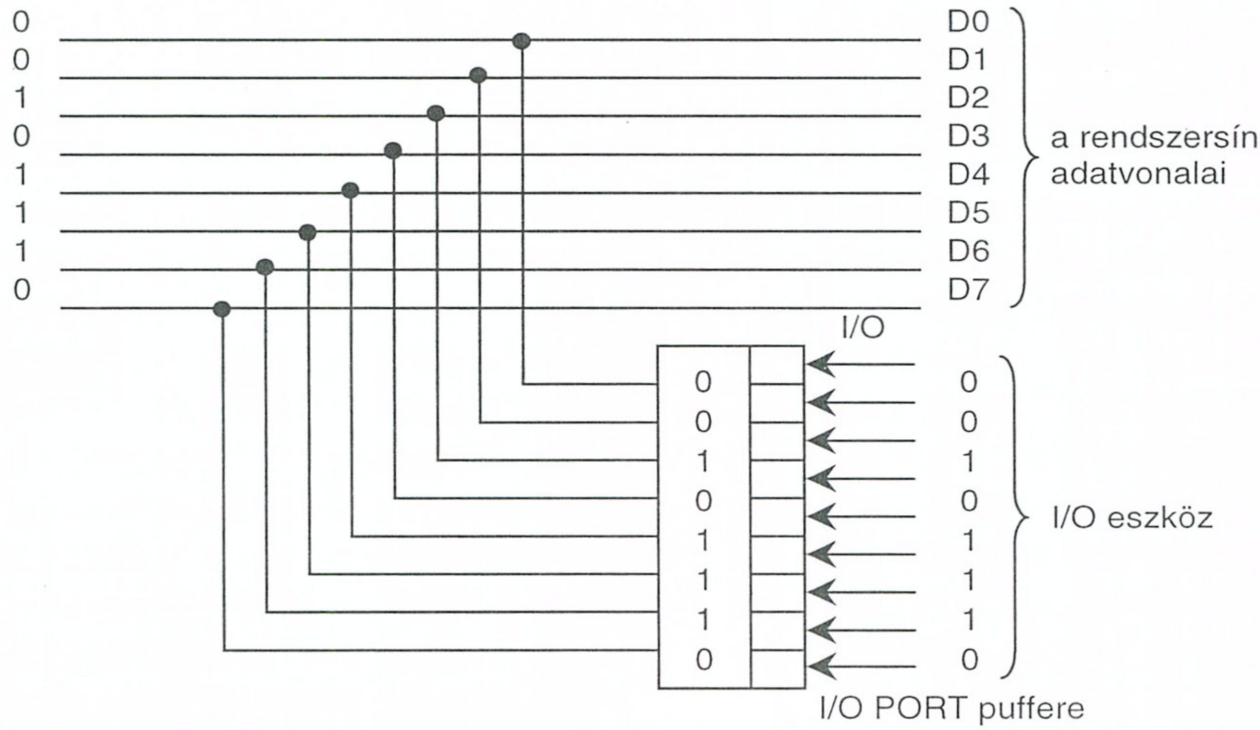
Az interfész két funkcionális egység összekapcsolhatóságát és együttműködését biztosító előírások összessége. Ezen előírások kiterjednek többek között – hardver esetén – a fizikai-mechanikai jellemzőkre (pl. csatlakozók), a definiált jelekre, azok elektromos jellemzőire, az egyes funkciókat realizáló jelfrekvenciákra, valamint – szoftvernél is – a definiált műveletek (pl. kapcsolatfelvétel, adatátvitel, szétkapcsolódás) megvalósítására.

Az interfész tehát két rendszer (vagy egy rendszer egyes egységei) közötti képzelt felületnek az a része, amelyen keresztül az adatok átvitele az igényelt illesztés biztosításával (kódátalakítás, sebességváltoztatás stb.) történik. Az interfészrendszer: kábelek, csatlakozók, meghajtó- és vevő-áramkörök, jelvezeték-előírások, időzítési és vezérlési egyezmények olyan készlete, amely biztosítja az egyes rendszertechnikai egységek közötti egyértelmű, rendezett és szervezett információcserét.

A PORT a mikroszámítógépnek olyan interfésze, amely a perifériális eszközökkel tart kapcsolatot. Ez biztosítja a szabványos csatlakozást a CPU és a perifériális egységek között, a rendszersín (külső busz) közbeiktatásával. Az I/O PORT pufferei a külső buszrendszer adatvezetékeihez csatlakoznak (IV.26. ábra).

Az I/O PORT nincs állandóan aktív kapcsolatban az adatvezetékekkel, mert általában azokon keresztül bonyolódik le például a CPU és a memóriák közötti adatforgalom is. A CPU az I/O eszközzel való műveletvégzés igényeinek megfelelő időben választja ki csak az I/O PORT-ot.

Amikor a külső I/O eszköz adatot továbbít a mikroszámítógépbe, azt az I/O puffer tárolja. Az adatátvitel ezután a IV.26. ábrán látható módon a sínrendszer segítségével valósul meg.



IV.26. sz. ábra: I/O PORT csatlakoztatása a rendszersín adatvezetékeihez és egy I/O eszközhöz

Az I/O PORT pufferek éppen úgy címezhetők a sín címvonalaival, mint a memóriarekeszek. Egyszerűen megoldható a címzés, ha kikötjük, hogy ha mikrogépünknek a legnagyobb helyi értékű címvezetéke (A15) logikai 0 értékű, akkor ez memóriamodult, ha logikai 1, akkor pedig I/O PORT puffert választ ki.

Sok mikroszámítógép rendszerben használnak különálló logikát az I/O PORT-ok kiválasztására, e célból két külön vezérlővonal csatlakozik a CPU-hoz. Ekkor

- Az IOSEL vezérlővonal jelszintje jelzi, hogy a címvezetékek az I/O PORT-puffer-kiválasztó kódot tartalmazzák.
- Az IORW vezérlővonallal az írás (WRITE) és az olvasás (READ) különböztethető meg. Ha az IORW logikai 1, akkor a külső adatbuszon lévő adatot az I/O PORT puffernek ki kell olvasnia, ha logikai 0, akkor a kiválasztott I/O PORT puffer tartalma kerül a külső adatbuszra.

Ha a mikroszámítógép kimenő jelei a külső I/O eszközök számára parancsokat továbbítanak, akkor ezeket I/O vezérlő jeleknek hívjuk.

A külső eszköz ugyanakkor a mikroszámítógép felé I/O állapotokat (I/O Status) jelző adatokat továbbít. Más szóval a mikroszámítógép processzorának kimenetei vezérlik a külső eszközt, ez viszont csak az állapotára vonatkozó információkat tudja eljuttatni a processzorhoz. Ez ezeket értelmezi, és így el tudja dönteni, hogy melyik I/O PORT-hoz kapcsolt eszköz van olyan állapotban, hogy az adatátvitelt végre lehet hajtani.

2.2. Az adatátvitel típusai

A mikroszámítógépek és a perifériák közötti adatátvitelt három fő módszerrel oldhatjuk meg:

- *Programozott* bemeneti/kimeneti (I/O) adatátvitelnél az összes adatátvitellel kapcsolatos műveletet a számítógép programja vezérli.
- *Megszakítással* kezdeményezett bemeneti/kimeneti (I/O) adatátvitelnél az adatátvitelt egy programmegszakítás előzi meg. Ekkor egy I/O eszköz „kényszeríti" a processzort, hogy félbeszakítsa az éppen futó program végrehajtását, majd az I/O eszköz kérését végrehajtsa.
- *Közvetlen memóriahozzáférés* esetén (DMA = Direct Memory Access) közvetlen adatátvitel jön létre a memória és a bemeneti/kimeneti (I/O) készülékek között a CPU igénybevétele nélkül.

2.3. Programozott adatátvitel (PIO mód)

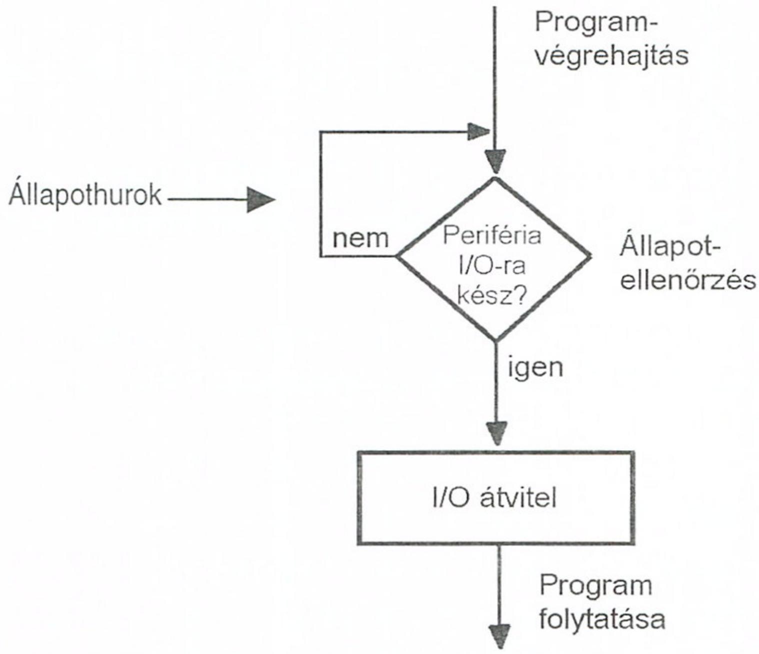
A programozott I/O adatátviteli műveletek végrehajtása általában a következő lépésekből áll:

- a periféria állapotát leíró adatok kiolvasása és vizsgálata;
- adatküldés a számítógépből a perifériához, vagy adatátvétel a perifériától.

Ehhez az I/O interfészek tartalmaznak egy állapotregisztert, mely a perifériák állapotára vonatkozó jelzőbiteket tárolja, és egy adatregisztert (puffert), melyen keresztül az adatcsere a perifériával megvalósulhat.

! Ha az adattovábbítás igénye az I/O eszköztől származik, az eszköz ezt azzal jelzi, hogy a megfelelő állapotbiteket az állapotregiszterben beállítja. Ha programmal kell megállapítani, hogy egy eszköz a műveletre készen áll-e vagy sem, akkor állapotregiszter bitjeit addig kell folyamatosan vizsgálni, míg abból az eszköznek az I/O művelet végrehajtására való alkalmassága meg nem állapítható.

Ez az állapotellenőrző hurok (azaz az állapotbitek sokszor ismételt ellenőrzése) drasztikusan leállítja a program érdemi részének végrehajtását, és így a hasznos processzoridő elfogadhatatlan pazarlását idézi elő (lásd IV.27. ábra).



IV.27. sz. ábra: Programvezérelt I/O adatátvitel; állapothurok

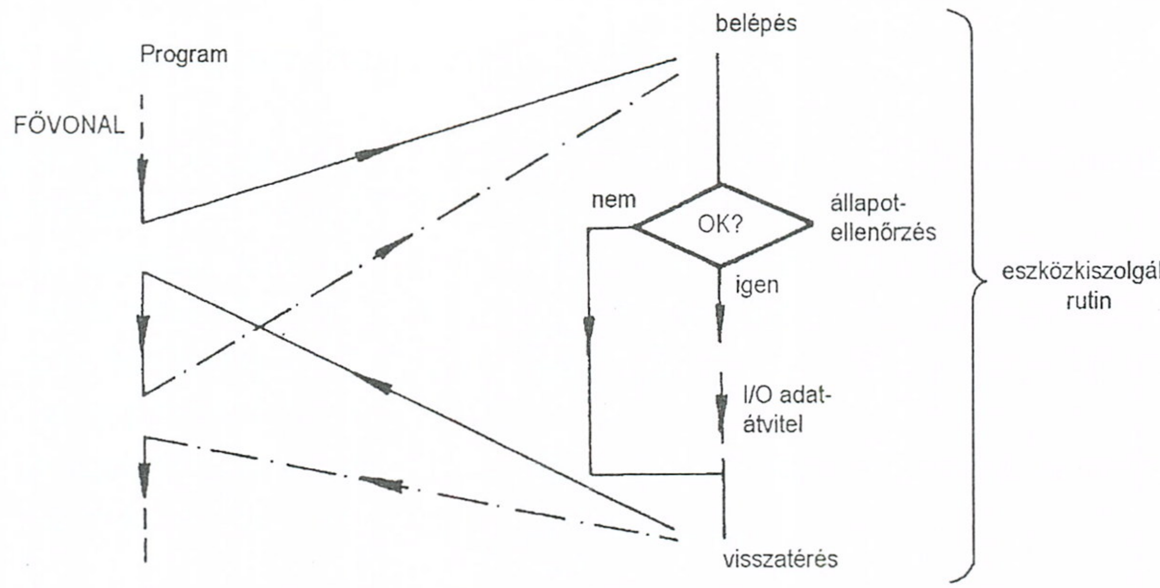
A processzoridő nem hatékony felhasználását különösen akkor érzékelhetjük, ha belegondolunk a működési sebességekben mutatkozó igen jelentős különbségekre. (Egyes perifériák több milliószor lassúbbak, mint a processzor.)

Ha például a processzor egy regiszter-regiszter műveletéhez szükséges időt 1 másodpercnek feleltetjük meg, akkor például a nyomtatás ehhez viszonyítva évekig tart. Így egy olyan helyzet áll elő az állapothurok esetében, hogy a processzor pár másodpercig dolgozik, azután például egy évig az állapotbiteket vizsgálja feleslegesen.

Mi lehet a megoldás erre a problémára?

Kézenfekvőnek látszik olyan algoritmust keresni, hogy a processzor az állapotbiteket ne folyamatosan ellenőrizze, hanem csak idő-

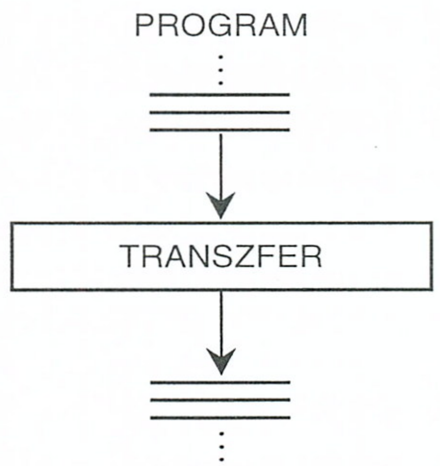
szakosan. Ha két ellenőrzés közötti időtartamot ügyesen úgy válasszuk meg, hogy az a periféria lassúságával összhangban legyen, akkor a processzor a két állapotellenőrzés közötti időt hasznos utasítások végrehajtásával ki tudja tölteni. Ezt a lehetőséget mutatja be a IV.28. sz. ábra.



IV.28. sz. ábra: Programvezérelt I/O adatátvitel a „beillesztés" művelettel

Az állapothurok problémájára akkor kell különösen odafigyelni, ha a processzor egyidejűleg több adatátvitelt igénylő eszközzel is kapcsolatban áll, mivel ekkor az időszakos állapotellenőrzéseket mindegyik eszközön végre kell hajtani.

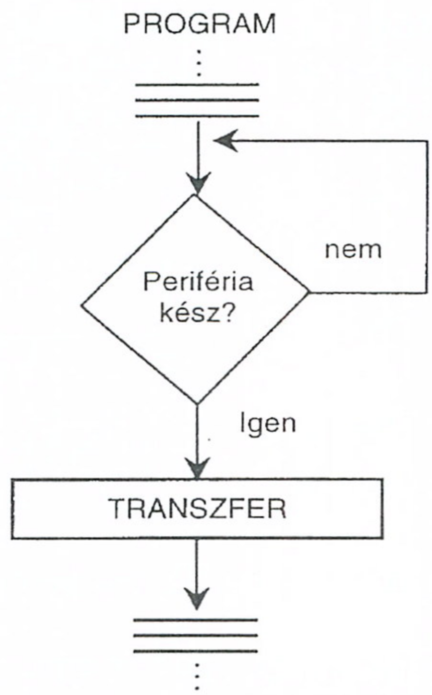
A programozott I/O adatátviteli művelet lehet feltétlen vagy feltételes (azaz valamilyen feltételtől függő). A feltétlen adatátvitel az adatok küldésére vagy fogadására mindig kész perifériát tételez fel (IV.29. ábra), ekkor tehát nem szükséges az állapotellenőrzés.



IV.29. sz. ábra: Feltétlen adatátvitel

A feltétlen adatátvitelre jó példa a számítógépen levő jelzőlámpák felgyújtása (LED kijelzők).

Feltételes adatátvitel esetén a CPU először megvizsgálja a periféria állapotregiszterét, és az adatközlés vagy vétel csak akkor következik be, ha a periféria arra készen áll (IV.30. ábra).



IV.30. sz. ábra: Feltételes adatátvitel

Ekkor alakulhat ki az előzőekben részletezett állapothurok.

2.4. Programmegszakítással (interrupt) történő adatátvitel

Az előzőekben láttuk, hogy a programvezérelt adatátvitel nem igazán hatékony megoldás, ha a CPU sebességét hatékonyan ki akarjuk használni. Ez az egyik oka annak, hogy a mikroszámítógépet a tervezők programmegszakítási lehetőséggel is ellátják.

Ekkor a számítógép rendelkezik egy olyan elektronikai részegységgel (megszakításvezérlő), amely lehetővé teszi az I/O eszközök számára, hogy I/O adatátviteli igényüket jelezzék a processzornak, ez pedig képes megszakítani a futó programot, és ezt követően kiszolgálni az I/O eszközt.

Mikor lehet szükség a mikroszámítógépünk programjának időszakos megszakítására?

A válasz egyszerű: amikor a jelenleg futó programnál a teljes rendszer hatékonyságának növelése miatt fontosabbá válik egy másik program végrehajtása.

2.4.1. A programmegszakítás okai

A programmegszakítás kérésének az I/O eszközök adatátviteli igénye mellett más oka is lehet, így például:

- a jelenleg futó program folytatása valamilyen akadály miatt nem lehetséges, az akadály elhárításához szoftver beavatkozás szükséges. (Például egy utasítás nullával akar osztani, vagy paritáshiba lépett fel);
- a számítógépet kezelő ember közölni kíván valamit a rendszerrel, például leüt egy billentyűt.

2.4.2. A megszakítás kiszolgálása

A megszakításkérés kiszolgálás során a következőket kell végrehajtania a CPU-nak:

- meg kell állítani a jelenleg futó programot;
- tárolni kell a jelenleg futó programra vonatkozó olyan adatokat, melyeket a megállított program későbbi folytatásához meg kell őrizni (például az utasításszámláló, és az állapotregiszter tartalma);
- el kell indítani a megszakítást kiszolgáló rutint, amely például elvégzi az I/O eszköz által igényelt adatátvitelt;
- be kell fejezni a megszakítást;
- vissza kell állítani az eredeti állapotot, vagyis a megszakított program folytatásához a processzort vissza kell állítani abba az állapotba, amelyben az a megszakítás időpontjában volt.

Programmegszakítást csak úgy lehet kiszolgálni, ha gondoskodunk azokról a hardver és szoftver eszközökről, melyek az előbbi lépések végrehajtását biztosítják.

2.4.3. A megszakítások típusai

A megszakítások a következő három fő típusba sorolhatók be:

- **külső megszakítások**, amelyeket egy vagy több I/O eszköz generál;
- **belső megszakítások**, amelyeket maga a mikroprocesszor hoz létre abból a célból, hogy sajátos feltételek vagy hibák előfordulását jelezze, például:
 - áramkimaradást,
 - a rendszer egyes elemeinek meghibásodását,
 - az adattovábbításban előforduló különböző hibákat stb.;
- **a szoftver által generált, szimulált megszakítások**, amelyek például elősegíthetik a programhibák behatárolását azáltal, hogy meghatározott címeken megszakítjuk a program futását és megjelenítjük a tároló tartalmakat (Ezt nyomkövetésnek, vagy DEBUG-nak nevezzük.)

A különböző megszakítási okok között fontossági sorrendet is fel kell állítani. Így például néhány megszakítást azonnal ki kell szolgálni, mások esetleg késhetnek, mert fontosabb feladat végrehajtása folyik. Így például a súlyos hardverhibákat jelző megszakításokat nyilvánvalóan azonnal ki kell szolgálni.

Ezért a megszakítási rendszernek prioritásuknak megfelelően különbséget kell tennie a különféle megszakítás források között, és a processzornak fontosságuk (prioritásuk) szerinti sorrendben kell végrehajtania a megszakítások kiszolgálását.

2.5. Adatátvitel közvetlen memóriahozzáféréssel (DMA)

A DMA vezérlő egy leegyszerűsített CPU-hoz hasonló hardver egység, amely az adatmozgatást közvetlenül irányítja a memória és az I/O készülék között. Ez lényegében úgy is felfogható, hogy az ebben résztvevő periféria közvetlen memóriahozzáféréssel rendelkezik.

A DMA alkalmazásának egyik célja a CPU mentesítése az adatátvitel közvetlen vezérlése alól. Emellett a DMA vezérléssel nagysebességű közvetlen adatátvitel érhető el a memória és az I/O készülékek között.

A CPU, az I/O készülékek, valamint a memória közös buszra vannak rákapcsolva, ezért a CPU és az I/O készülékek azonos ciklusban nem férhetnek hozzá a memóriához. Emiatt valamilyen formában a CPU és a DMA vezérlő között meg kell osztani a sín használatát.

Ennek megoldási formája szerint különböző DMA adatátviteli eljárásokat különböztethetünk meg. Ezek: a CPU leállítása (CPU halt), a memória-időszelet eljárás és a cikluslopás.

2.5.1. DMA adatátviteli eljárások típusai

A **CPU leállítási eljárásnál** a DMA kérésére a CPU leáll és lekapcsolódik a buszról a DMA adatátvitel tartama alatt. Ez a módszer lassítja a CPU működését, bár a DMA átvitel nyilván ebben az esetben a leggyorsabb.

A **memória-időszelet eljárásnál** a memóriaciklust két részre bontják fel: az egyik a CPU-é, a másik a DMA-é. Ez a módszer nagy CPU végrehajtási és nagy DMA adatátviteli sebességet eredményez, mivel CPU és DMA memóriahozzáférés minden ciklusban van. Hátránya, hogy megvalósítása igen nagy sebességű, drága memóriát igényel.

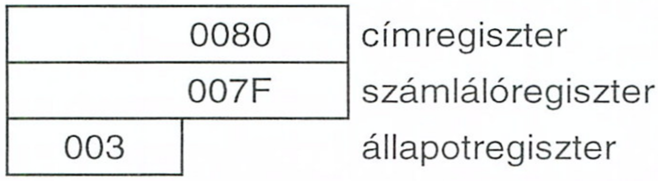
A **cikluslopásos eljárás** kompromisszum az előző kettő között, ekkor a CPU és a DMA vezérlő átlapolva használja a buszt. Ha a CPU-nak és a DMA-nak azonos időben lenne szüksége memóriára, akkor a DMA-nak prioritása van a CPU-val szemben, és a CPU addig vár, amíg a DMA ciklus be nem fejeződik. Ez a CPU működését lassítja, de nem állítja le, ezért ez a megoldás a legtöbb esetben kielégítő.

2.5.2. A DMA regiszterei

Egy DMA vezérlő a következő három regisztert használja:

- a *címregisztert*, amely azt a memóriarekesz-címet tárolja, amely a következő írási/olvasási műveletben részt vesz;
- a *számlálóregisztert*, amely az átvitt memóriarekeszeket számlálja;
- az *állapot- (status) regisztert*, amelynek tartalma meghatározza az adatáramlás irányát, vagyis a DMA üzemmódját.

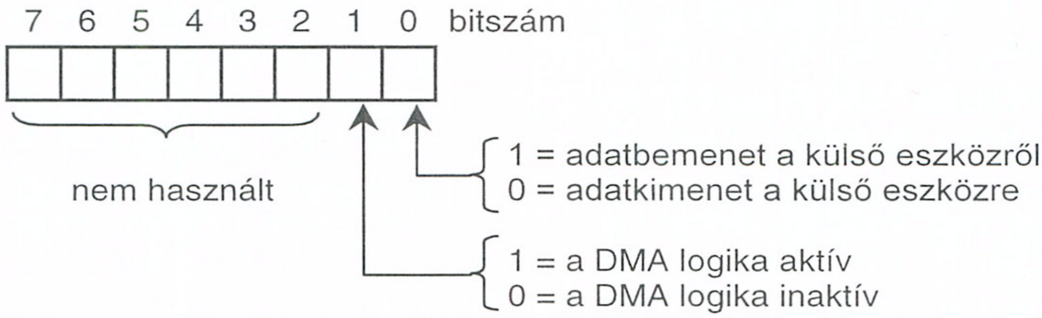
A CPU szempontjából nézve ezek a regiszterek úgy kezelhetők, mint az I/O PORT-ok vagy a címezhető memóriarekeszek. A DMA adatátvitel megkezdése előtt az átvitelhez szükséges adatokkal a CPU a cím-, a számláló- és az állapot-regisztereket feltölti (IV.31. ábra). Ezt követően engedélyezi a CPU a DMA átvitel megkezdését.



IV.31. sz. ábra: A DMA regisztereinek beállítása (példa)

A IV.31. ábrából leolvasható, hogy az átviendő bájtszám $007F_{16}$ (vagyis, hogy az adatpuffer $7F_{(16)}$ bájt hosszú) és a külső eszközről érkező adat elsőként a $0080_{(16)}$ című memóriarekeszbe kerül tárolásra.

A IV.32. ábrán mutatjuk be az állapotregisztert, vagyis a hardver üzemmód beállítását. Azzal, hogy a $03_{(16)}$ érték az állapotregiszterbe kerül, meghatározottá válik, hogy külső eszköz adata kerül a bemenetre és a DMA eszköz aktivizálódik.



IV.32. sz. ábra: A DMA az általa kezdeményezett műveleteket a CPU kezdeti segítségével hajtja végre. A CPU betölti az adatokat a DMA regisztereibe, vagyis a DMA vezérlő működési feltételeit a CPU állítja be, kiküldi az olvasási parancsot, az induló memóriacímet, valamint a szószámra és a memóriacímre vonatkozó adatokat.

A DMA művelethez a következő lépéseket kell végrehajtani:

1. Be kell tölteni az induló címet a DMA címregiszterébe.
2. Be kell tölteni az átviendő rekeszszámra vonatkozó adatokat a DMA számlálóregiszterébe.
3. Be kell tölteni egy vezérlőkódot (példánkban a $03_{(16)}$-at) a DMA állapotregiszterébe. A vezérlőkód meghatározza az adatátvitel irányát és annak megfelelően állítja be a DMA-t.

2.5.3. A DMA művelet végrehajtása

Vizsgáljuk meg, mi történik, ha DMA lehetőséget építünk be a mikroszámítógéprendszerbe. Amikor a DMA vezérlő egy adatszót fogad az I/O interfésztől, kéri a rendszerbusz használatát a DMA átvitelhez. Amint megkapta a buszt; kiküldi a memóriarekesz címét és az adatrekesz tartalmát. A "memória kész" jel vételkor a DMA vezérlő megvizsgálja a számlálóregiszterben a rekeszszámot. Ha ez nem egyenlő zérussal, akkor a számlálóregiszter tartalmát 1-gyel csökkenti, a címregiszter tartalmát pedig növeli. Ennek hatására a periféria pufferből a memóriába egy újabb adat átvitele megy végbe. Ha a számlálóregiszter tartalma eléri a zérust, a DMA vezérlő befejezi az adatátvitelt és közli a CPU-val, hogy az adatátvitel kész.

2.5.4. DMA vezérlőfunkciók

Egy DMA adatátvitel végrehajtásához a következő vezérlési feladatokat kell ellátni:

Címvonalvezérlés: a DMA-t tartalmazó rendszerekben a memóriacímbuszt vagy a CPU, vagy a DMA hajtja meg attól függően, hogy az adott ciklusban a memóriát melyik eszköz használja. A DMA ciklusban a DMA vezérlőnek a kívánt DMA művelet elvégzéséhez szükséges címet kell a címbuszra adni.

Adatátviteli vezérlés: a DMA vezérlőnek a memória és az I/O készülék közötti közvetlen adatátvitelhez - megfelelő időzítéssel - vezérlőjeleket kell szolgáltatni. E vezérlőjelek csak a DMA ciklusban kapcsolódnak a vezérlőbuszra.

Címtárolás: a DMA vezérlő címregisztere tartalmazza a következő írásra vagy olvasásra kerülő adat címét. Ezt minden átvitel után inkrementálni vagy dekrementálni kell.

Adatszámlálás: a DMA adatátvitel indításakor a CPU betölti a DMA vezérlő számlálóregiszterébe az átvitelre kerülő rekeszek számát. A DMA adatátvitel alatt a DMA vezérlő számlálja az átvitt rekeszeket és a megadott számú adat átvitele után befejezi az adatátvitelt.

Üzemmódvezérlés: a DMA hardver üzemmódvezérlő regiszterét a CPU tölti fel az adatátvitel előtt.

A DMA vezérlő helye szerint két megoldást különböztetünk meg:

- Elosztott DMA: az I/O készülékekben van a DMA vezérlő.
- Centralizált DMA: egyetlen DMA vezérlő szolgál valamennyi I/O készülék vezérlésére.

2.6. Külső buszrendszer (system bus)

A mikroszámítógép-rendszer sínrendszere kapcsolja össze a CPU-t az interfészeken keresztül a perifériákkal. A külső buszrendszer logikailag

- adatvezetékekből vagy adatsínből (32 vagy 64 bit egyidejű átvitelére),
- címvezetékekből vagy címsínből (általában 32 bit egyidejű átvitelére),
- vezérlővezetékekből vagy vezérlősínből áll.

Az adat- és címvezetékek száma a belső és külső buszon általában azonos, napjainkban a legtöbb mikroszámítógép 32 címvonallal rendelkezik.

A vezérlősín a mikroszámítógép részegységei közötti vezérlőadatok átvitelét biztosítja. Ezek lehetnek

- I/O eszközvezérlő jelek
- megszakítási rendszer vezérlőjelei
- DMA vezérlőjelei
- sínvezérlőjelek, melyek például a sínhasználat kérésére és visszaigazolására szolgálnak.

A tervezések során két véglet figyelhető meg a CPU-val megvalósított rendszerekben:

- Az egyik véglet szerinti megoldásnál "okos" CPU-t és "buta" perifériákat használnak, vagyis a vezérlést teljes egészében a CPU hajtja végre. A CPU mondja meg a perifériáknak, hogy mit kell tenniük és a perifériák szolgaian végrehajtják az utasításokat.
- A másik véglet feltételezi, hogy a csatlakozó készülékek, perifériák jelentős mennyiségű belső logikát tartalmaznak. Így a CPU-nak csak elemi vezérlőjeleket kell kiadnia, amelyeket a csatlakozó készülékek értelmeznek és "saját fejük" szerint hajtanak végre.

Amikor a CPU teljes mértékben ural egy rendszert, akkor a hozzá csatlakoztatott készülékek számos vezérlőjelet fogadnak, amelyek értelmezik az adatbuszon végbemenő eseményeket. A másik véglet, amikor a CPU csak két vezérlőjelet ad ki. Az egyik jelzi, hogy I/O (input/output) vagy memóriaművelet van-e folyamatban, a másik az adatforgalom irányát jelzi, vagyis, hogy adatbevitel vagy adatkihozatal történik-e.

2.7. Soros adatátvitel

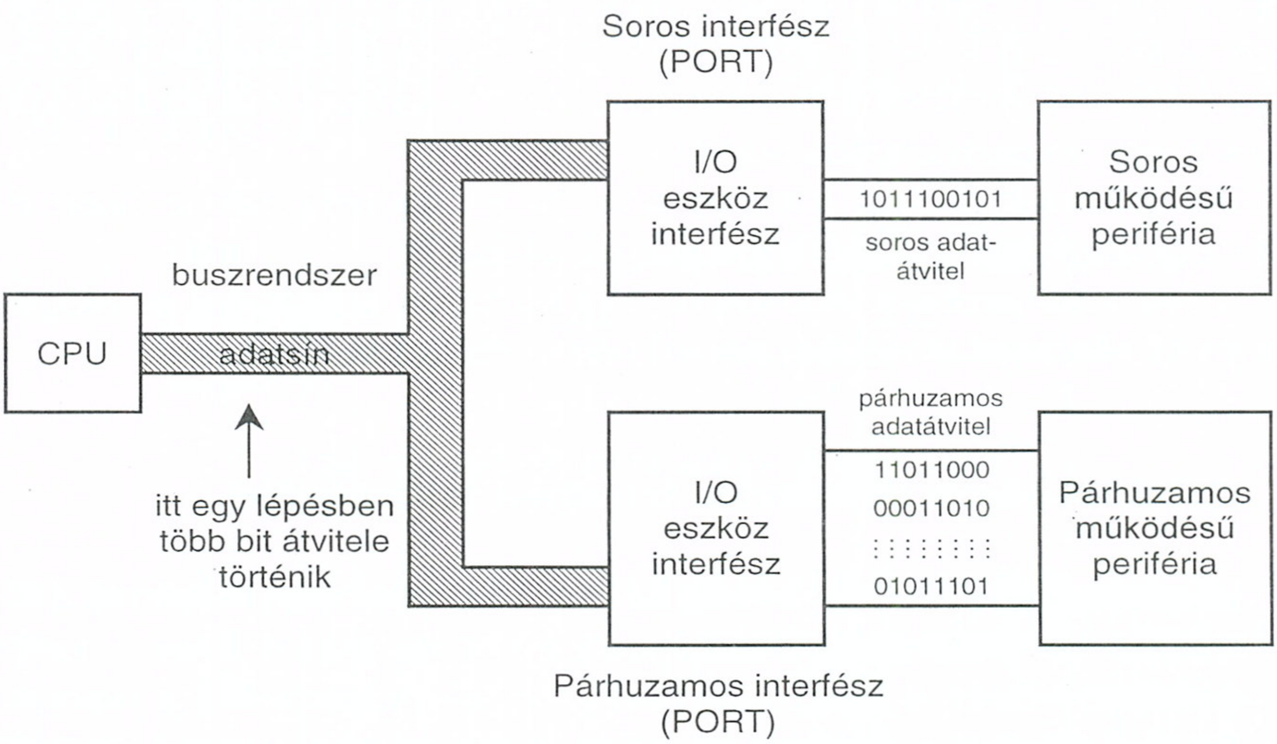
2.7.1. A soros és párhuzamos adatátvitel

Az előzőekben a CPU és a periféria interfészek közötti adatátvitel megszervezésének kérdéseivel és lehetséges formáival foglalkoztunk.

Hogyan történhet az adatátvitel a periféria interfészek és a perifériák között?

Erre két, lényegében különböző megoldás létezhet:

- a periféria interfész és a periféria között az adatokat bitenként sorba egymás után visszük át, ezt **soros adatátvitel**nek nevezzük, !
- a periféria interfész és a periféria között az adatokat bitcsoportonként egyszerre visszük át, ezt **párhuzamos adatátvitel**nek nevezzük. !



IV.33. sz. ábra: Soros és párhuzamos adatátvitel

A soros működésű I/O eszköz interfészt soros, a párhuzamos működésűt pedig párhuzamos portnak is szokták nevezni.

Nyilvánvalóan a párhuzamos adatátvitelnek megvan az az előnye, hogy gyorsabb a sorosnál, mivel egy lépésben egy bitcsoportot viszünk át. Ehhez viszont legalább annyi vezetéket (adatutat) kell biztosítani az adatok átviteléhez, mint ahány bitből áll az átvitt bitcsoport.

A soros átvitelhez viszont szélsőséges esetben egy vezetékpár is elegendő az adatátviteli összeköttetéshez, ami jelentősen csökkentheti a költségeket.

Az előbbieket figyelembe véve a párhuzamos adatátvitelt csak a számítógép közelébe elhelyezhető perifériák, így például nyomtató esetében szokták alkalmazni.

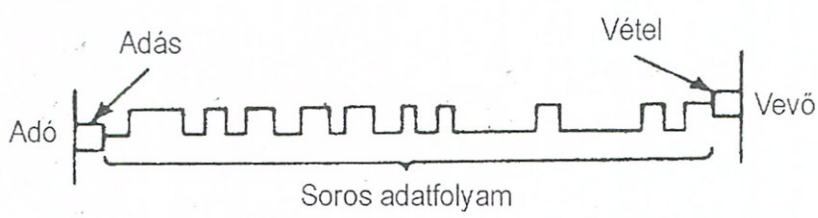
Az elmúlt évek technikai fejlődésének eredményeként egyre nagyobb sebességű soros adatátvitel megvalósítására nyílt lehetőség, így a párhuzamos adatátvitel gyorsasága ma már nem jelent előnyt. Erre példa, hogy a mikroszámítógépek körében gyorsan terjed az úgynevezett univerzális soros buszrendszer (USD = Universal Serial Bus), mely az összes kis és közepes teljesítményű periféria csatlakoztatását biztosítja a számítógéphez egy nagysebességű soros adatátvitellel. Az USB a jövőben várhatóan kiváltja a mai számítógépek soros és párhuzamos portját.

Az előbbiek mellett a soros adatátvitelt alkalmazzák a távadatfeldolgozás során a számítógéphálózatokban is.

Ezek a tényezők is indokolják, hogy a soros adatátvitel alapfogalmait és eljárásait kissé részletesebben is megvizsgáljuk.

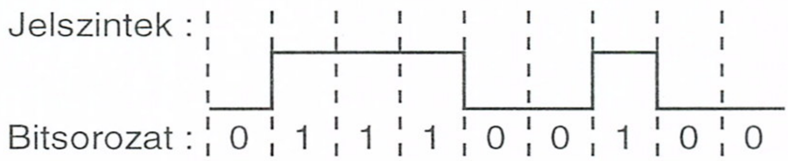
2.7.2. A soros adatátvitel bitjeinek felismerése

A soros adatátvitelben mindig két eszköz vesz részt, egyik az *adó*, a másik a *vevő* szerepkörét tölti be.



IV.34. sz. ábra: Adó-vevő csatlakozó pontok

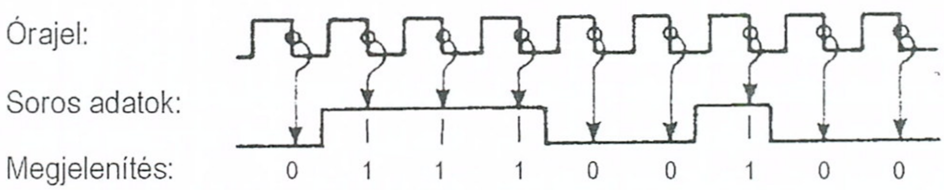
A soros adatfolyamban bitsorozatot viszünk át, minden bitet a jel feszültségszintjével adunk meg. Például, ha az átvitt bitsorozat 011100100, akkor ez jelszintekben a következő ábra szerint jellemezhető:



IV.35. sz. ábra: A soros adatfolyam jelszintjei

Vegyük észre, hogy az ábrán a jelszintek értelmezése csak úgy lehetséges, ha ismerjük egy bit időtartamát. Például, ha a bitek időtartama az ábrán láthatónak csak a fele lenne (azaz a szaggatott függőleges vonalak kétszer sűrűbben helyezkednének el), akkor a soros adatfolyamot 001111110000110000 birsorozatnak értelmeznénk.

Ezért a soros adatfolyam értelmezéséhez a vevőnek fel kell ismernie az adatbitek határait. Erre a célra órajelet alkalmazhatunk a IV. 36. ábra szerint.



IV.36. sz. ábra: A soros adatok értelmezése

Az ábra mutatja, hogy az órajel lefelé menő éle az az időpillanat, amikor a vevő mintát vesz a soros adatok jelszintjéből és eszerint értelmezi a bitsorozatot. Az is nyilvánvaló, hogy a vevő órajelekkel értelmezi a soros adatfolyamot, akkor az adónak is azonos frekvenciájú órajelekkel kell létrehozni a soros adatok jelszintjét.

Megjegyzés: Nem szabad összetéveszteni a soros adatátvitel szinkronizálásához szükséges órajelet a processzor órajelével, a kettő között több tízezerszeres is lehet a sebességkülönbség, nyilvánvalóan a processzor javára.

2.7.3. Az adó és vevő szinkronizálása

Láttuk, hogy a soros adatfolyam értelmezéséhez az szükséges, hogy az adó és a vevő azonos frekvenciájú órajelet alkalmazzon. Most már csak az a kérdés, hogy hogyan lehet azt biztosítani, hogy az órajelek generálását a vevő az adóval összehangoltan képezze és például a két órajel ne „csússzon" el egymástól. Erre az egyik megoldás az lehet, hogy az adatok átvitelét szolgáló jelvezeték mellett még egy vezetéket alkalmazunk, melyen az órajelet is átvisszük az adó és a vevő között. Ez viszont nyilvánvalóan nem a leggazdaságosabb.

Hatékonyabb az az eljárás, amelyben az átvitt bitsorozatot használjuk fel az adó és vevő órajelének összehangolásához, szinkronba hozásához. Ez az jelenti, hogy előírunk egy speciális bitsorozatot, amit szinkronizáló jelnek nevezünk, aminek feladata az adó és vevő működésének szinkronizálása, az órajelképzés időbeli összehangolása.

A szinkronizáláshoz megfelelő szabályrendszert is ki kell alakítani, például azt, hogy a szinkronizáló bitmintának mindig meg kell előznie az érdemi adatbiteket. Ez vezet el a kommunikációs protokoll fogalmához.

2.7.4. Kommunikációs protokoll

Az adatátviteli szabályok, megállapodások összességét kommunikációs protokollnak nevezzük. Ez mint láttuk tartalmazza az adó és vevő oldal szinkronizálásához szükséges szabályokat, de ennél jóval többre is kiterjed. Így például ide tartoznak

- azok az eljárások, melyekkel az adó jelzi, hogy adatot kíván küldeni, a vevő pedig visszaigazolja, hogy képes fogadni az adatokat,
- az adatátvitel során fellépő hibák felismerési, illetve kijavítási szabályai.

A soros adatvonalakat – szemben a párhuzamos adatátvitellel – nem kísérik minden esetben olyan vezérlővonalak, melyekkel az adatátvitel vezérléséhez szükséges összes információ az adó és vevő között átvihető lenne. Ezért minden soros adatátvitelhez egy kommunikációs protokollra van szükség, amely a vevő és az adó összehangolt működését biztosítja.

A kommunikációs protokollok egyrészt szabványok, másrészt egy konkrét számítógépnél különböző hardver és szoftver elemek, melyekben a szabványban megfogalmazott szabályrendszer ölt testet.

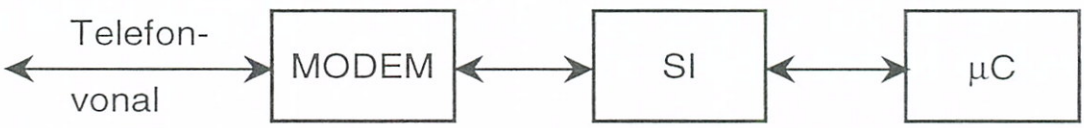
Így például a soros porton keresztül történő soros adatátvitel szabványa az Electronic Industries Association nevű szakmai szervezet által kialakított RS-232-C. Ez részletesen meghatározza az adatátvitel mechanikai, villamos, funkcionális és eljárási szabályrendszerét.

Egy kommunikációs protokoll nem csak néhány egyszerűen megfogalmazható szabály együttese. Például ezek legtöbbjének műszaki dokumentációja csak több ezer oldalban fogalmazható meg.

2.7.5. Soros adatátvitel távbeszélő vonalakon

Mivel a normál telefonvonalak a jeleket analóg formában viszik át, a soros adatátvitelnél viszont bináris digitek formájában adottak az adatok, ezért a távbeszélővonalakon történő adattovábbításhoz speciális eszközre, az úgynevezett MODEM-re is szükség van.

A MODEM (MODULÁTOR/DEMODULÁTOR) egy analóg-digitális és digitális-analóg konverziót megvalósító eszköz, amely a telefonvonal analóg jeleit digitális bitsorozattá alakítja át, illetve a digitális bitsorozatot a telefonvonalnak megfelelő analóg jelekké fordítja át. Ennek megfelelően a modemet közvetlenül a telefonvonal és a számítógép soros interfésze közé kell elhelyezni.



IV.37. sz. ábra: Adattovábbítás a távbeszélő hálózat és a számítógép között (SI = soros interfész és μC = mikroszámítógép)

A távbeszélőhálózat adatátviteli vizsgálatánál az első szempont annak a ténynek a szem előtt tartása, hogy az adatátvitelben résztvevő készülékek állandóan váltogatják szerepüket: egyszer adók, máskor vevők. Ez teljesen hasonló egy normál telefonbeszélgetéshez, ahol a beszélő az adó és a hallgató a vevő.

Az adatátvitel iránya és egyidejűsége szerint egy összeköttetés lehet:

- szimplex (simplex), ha az adatátvitel csak egy irányban folyhat, azaz az adó és vevő szerepe nem változhat,

- fél-duplex (half duplex), ha az adatokat egy vezetéken mindkét irányba továbbíthatjuk, de az adatátvitel egy adott időben csak az egyik irányban folyik. Ebben az esetben tehát az adó és vevő időnként szerepet cserél,
- teljes duplex (full duplex), amikor az adatokat egy időben mindkét irányban továbbítani lehet. Ez két vezetéket tételez fel, és az adatok továbbítása ekkor mindkét irányban párhuzamosan történik.

A telefonvonalon történő adattovábbítás során az adatok meghibásodhatnak, például a bitsorozatban egyes bitek ellenkezőjükre változhatnak. Ezért fontosak azok az eljárások, melyekkel a vevőkészülék képes ezeket a hibákat felismerni, illetve kijavítani. A hibafelismerésre használható a már megismert paritásbit ellenőrzés. Léteznek olyan eljárások is, melyekkel a hibák nemcsak felismerhetők, de azok egy része ki is javítható.

A soros adatátvitelnél az adatok tömörítése is fontos, mert ha sikerül az átviendő adatokat jelentősen összetömöríteni, akkor (mivel kevesebb mennyiségű bitet kell átvinni) az adatátvitel kisebb teljesítményű összeköttetések esetén is gyors lehet.

A modemek szabványos hibajavító és tömörítő eljárásai az MNP (Microcom Networking Protocol) protokollok.

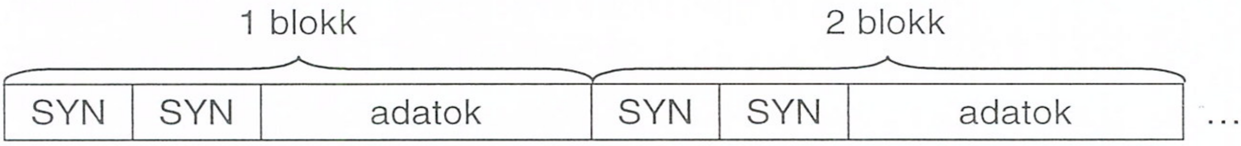
Az adatátvitelnek fontos jellemzője a sebessége, melyet Baud-ban mérünk. 1 Baud sebességű az az átvitel, mely 1 másodperc alatt 1 jelet visz át. Ha az átvitt jelek bináris értékek, akkor a Baud megegyezik a bit/szekundummal.

2.7.6. Szinkron és aszinkron soros adatátvitel

A soros adatátvitel két alaptípusa a szinkron és az aszinkron átvitel.

A **szinkron adatátvitelnél** az adó és vevő szinkronizálását a SYN bitcsoport = 01111110 érzékelése biztosítja a vevő részéről.

Az átviendő érdemi adatokat mindig két SYN jel vezeti be, az átvitt bitek mennyisége rögzített, azaz ezek blokkszervezésűek.



IV.38. sz. ábra: Szinkron soros adatátvitel

A szinkron adatátvitel órajellel vezérelt, azaz az egyes adatokhoz tartozó jelek csak egy meghatározott alapidőtartam egészszámú többszörösei lehetnek.

Az **aszinkron adatátvitelnél** az adatok elejét és végét két speciális jel a START és STOP jel jelzi, az ezek között elhelyezkedő jelek értelmezendők adatként. Ebben az esetben az átvitt bitek mennyisége változó is lehet.

START	adatok	STOP	SZÜNET	START	adatok	STOP

IV.39. sz. ábra: Aszinkron soros adatátvitel

Az aszinkron átvitelnél nem szükségképpen van folyamatos kapcsolat az adó és vevő között, ezek szinkronban csak az adatátvitel ideje alatt vannak. A szinkronizálást a START jel eredményezi.

Az adatátviteli eljárások lehetnek karakterorientáltak, ekkor az átvitel egysége az egy karakter kódjának megfelelő bitszám (ez még a távadatfeldolgozás kezdeteitől származtatható, amikor a karakterekből álló szöveges információ átvitele volt a jellemző).

A hálózatok, a hang- és képátvitel elterjedésével kerültek előtérbe a bitorientált eljárások, melyek már nemcsak a karakterkódoknak megfelelő számú bitsorozatok lépésenkénti átvitelét is biztosítják, hanem nagyobb mennyiségű bit (például képek) is továbbítható távadatfeldolgozással.

Ellenőrző kérdések

1. Milyen feladata van a mikroprocesszornak?
2. Milyen funkcionális egységei vannak a mikroprocesszornak?
3. Mi a regiszter?
4. Mi a különbség a rendszer- és az általános regiszterek között?
5. Milyen tipikus regisztereket ismer a processzorban?
6. Mit nevezünk elérési időnek?
7. Mit tartalmaz az utasításregiszter?
8. Mit tartalmaz az utasításszámláló?
9. Mi a pufferregiszter feladata?
10. Milyen elemi lépésekből áll egy bináris összeadás?
11. Mi a feladata az ALU-nak?

12. Milyen alapműveletek elvégzésére alkalmas az ALU?
13. Milyen részei vannak az ALU-nak?
14. Mi a feladata a vezérlőegységnek?
15. Mit jelent az utasításkód értelmezése?
16. Sorolja fel néhány állapotinformációs bit jelentését!
17. Mi a funkciója a jelzőbitnek (flag)?
18. Mit nevezünk a processzor utasításkészletének?
19. Mit jelent a gépi utasítás szerkezete?
20. Milyen részekből áll a gépi utasítás?
21. Mit határoz meg a műveleti kód a gépi utasításban?
22. Mit tartalmaz a gépi utasítás címrésze?
23. Milyen címzési eljárásokat ismer?
24. Mit jelent az abszolút címzés?
25. Mit tartalmaz a gépi utasítás címrésze relatív címzés esetén?
26. Mi a közvetlen adatcímzés lényege?
27. Mi jellemzi a vermet?
28. Jellemezze a LIFO szervezésű memóriát!
29. Mit nevezünk veremmutatónak?
30. Milyen veremműveletek vannak?
31. Jellemezze a POP utasítást?
32. Mi a hatása a PUSH műveletnek?
33. Hogyan működik a kaszkád verem?
34. Mit nevezünk szubrutinnak?
35. Mi történik a CALL szubrutinhívó utasítás hatására?
36. Hogyan lehet a szubrutinból visszatérni?
37. Mit jelent a rekurzív szubrutin?
38. Mit jelent a közvetett címzés?
39. Hogyan kapjuk meg a valódi címet indexelt címzésnél?
40. Tipikusan mire használható az indexelt címzés?
41. Milyen részekből épül fel a mikroszámítógép?
42. Mi a ROM?
43. Milyen csatlakozóhelyei vannak egy ROM chipnek?
44. Mivel kapcsolható össze a processzor a memóriachipekkel?
45. Mi a RAM?
46. Milyen csatlakozóhelyei vannak egy RAM chipnek?
47. Mi a különbség a ROM és RAM chip csatlakozóhelyei között?
48. Egy 8-bites rekesz adatainak tárolására hány RAM chip szükséges? Indokolja meg a választ!
49. Mit értünk interfészen?
50. Milyen fizikai elemekből épül fel az interfész?
51. Mit nevezünk PORT-nak?
52. Mihez csatlakoznak áramkörileg az I/O PORT pufferei?

53. Milyen lehetőségek vannak az I/O PORT címzésére?
54. Mire használható a CPU számára az I/O eszköz állapotinformációja?
55. A CPU és a perifériák közötti adatátvitelnek milyen fontosabb típusai vannak?
56. Milyen lépésekből áll a programozott I/O adatátvitel?
57. Az állapotellenőrzés feladata milyen problémákat okoz a processzoridő hatékony felhasználásában?
58. Mondjon példát a feltétlen adatátvitelre a CPU és a perifériák között!
59. Milyen okai lehetnek a programmegszakításnak?
60. Milyen feladatokat kell megoldani a megszakítás-kiszolgálás során?
61. Milyen típusai vannak a megszakításoknak?
62. Miért van a megszakítási okoknak prioritása?
63. Mi a DMA vezérlő?
64. Milyen típusai vannak a DMA adatátviteli eljárásoknak?
65. Mi a lényege a memória–időszelet eljárásnak?
66. Mit jelent a cikluslopás?
67. Milyen feladatokat ellátó regisztereket tartalmaz a DMA vezérlő?
68. Hogyan történik meg egy DMA művelet végrehajtása?
69. Milyen vezérlőfunkciói vannak a DMA-nak?
70. Milyen részekből épül fel a külső (rendszer) busz?
71. Milyen jellegű vezérlőinformációkat viszünk át a buszrendszeren?
72. A CPU és a perifériák közötti munkamegosztásnak milyen szélsőséges esetei vannak?
73. Mit jelent a soros adatátvitel?
74. Mit nevezünk párhuzamos adatátvitelnek?
75. Miben különbözik egymástól a soros és párhuzamos port?
76. Milyen perifériákat kapcsolunk a párhuzamos porthoz?
77. Mi az USB?
78. Milyen adatátvitelt alkalmazunk a távadatfeldolgozás során?
79. Hogyan ismerjük fel soros adatátvitelnél az adatbitek határait?
80. Hogyan szinkronizálhatjuk az adó és vevő működését?
81. Mit nevezünk kommunikációs protokollnak?
82. Soroljon fel legalább két területet, melyet a kommunikációs protokollban szabályozni kell!
83. Mi a modem?
84. Mit értünk szimplex összeköttetésen?
85. Mit értünk fél-duplex összeköttetésen?

86. Mit értünk duplex összeköttetésen?
87. Mi a szerepe az MNP protokollnak?
88. Mi az adatátvitel sebességének mértékegysége?
89. Jellemezze a szinkron adatátvitelt!
90. Jellemezze az aszinkron adatátvitelt!
91. Mikor szinkronizált az aszinkron átvitel?

V. fejezet

SZOFTVER ALAPISMERETEK

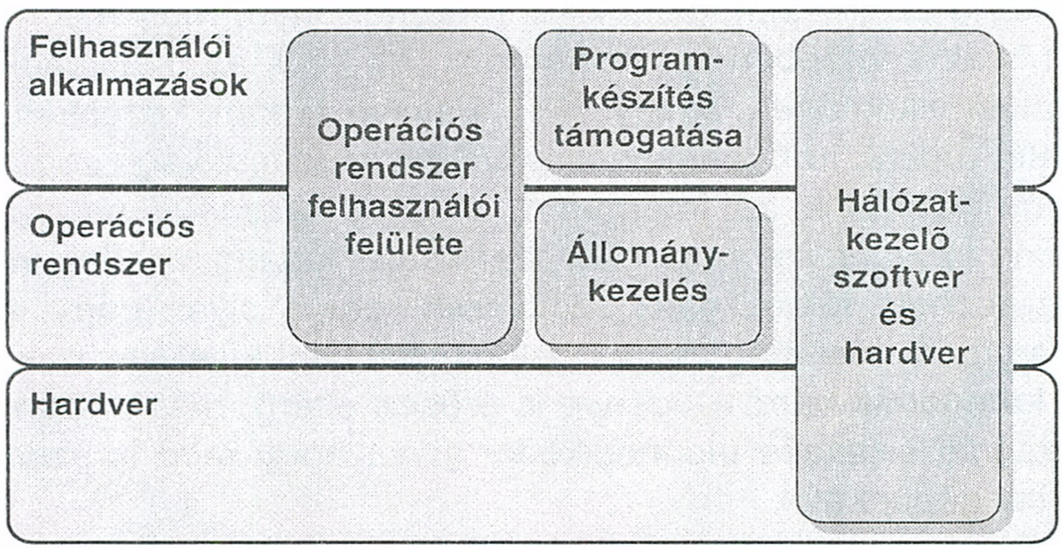
1. Bevezető

Mint az előbbiekben láttuk, a mikroprocesszor alapú rendszer egy olyan jól felkészült automata, amely precízen, gyorsan, de szolgaian hajtja végre azokat az utasításokat (programokat), melyeket memóriájában talál. Legtöbben azt szeretnénk, ha ezek az utasítások olyan algoritmust alkotnának, amelyekkel adatokat tudunk kezelni, válogatni, levelet tudunk írni, grafikonokat tudunk készíteni vagy segítségükkel barangolhatunk az interneten, azaz *felhasználói programokat*, vagy más szóval *alkalmazásokat* szeretnénk futtatni, esetleg írni.
A beszerezhető felhasználói szoftverek száma olyan nagy, hogy a teljesség igényével még csoportosításukkal sem kísérletezhetünk. Az egyes kategóriák hardver igénye is erősen eltérő, már a számítógép vásárlásánál érdemes elgondolkodni azon, hogy mire is szeretnénk használni gépünket.

Egy mai átlagos munkahelyen a következő szoftverekkel találkozhatunk:

- **Operációs rendszer**, pl. Windows 95/98/NT, DOS, Linux, OS/2
- **Irodai szoftvercsomag** (szövegszerkesztő, táblázatkezelő, adatbáziskezelő, prezentációs grafika), pl. Microsoft Office
- **Ügyviteli rendszerek** (bérszámfejtés, raktárnyilvántartás, könyvelői rendszerek), pl. MaxBer, COBRA stb.
- **Tervező rendszerek** (CAD – Computer Aided Design), pl. PCAD, ArchiCAD, MathCAD
- **Programfejlesztést támogató rendszerek**, pl. Turbo Pascal, C, Java Builder, Visual Basic, Negyedik generációs nyelvek
- **Számítógép hálózattal kapcsolatos szoftverek**, pl. Pegasus Mail, Netscape, Internet Explorer
- **Segédprogramok** (tömörítő, víruskereső stb.), pl. PkZip, F-Prot

Ebből a bő kínálatból azokat az elemeket válogattuk ki, melyek minden alkalmazás készítésénél vagy működtetésénél szerepet játszanak. (Ez a csoportosítás egyáltalán nem akar egyes szoftvereket mások fölé vagy alá rendelni.) Az következő ábra, amely a szerencsés kiemelt területeket helyezi el a számítógép-rendszer egészében talán annak illusztrálására is alkalmas, hogy nem is mindig olyan egyszerű meghúzni a határokat a felhasználói szoftver, az operációs rendszer és a hardver között.



V.1. sz. ábra
A felhasználó alkalmazások, az operációs rendszer és a hardver összefüggései

Programfejlesztési támogatás

Alkalmazásainkat elkészíthetjük akár gépi kódban is, de ez igencsak fárasztó, és a számítógépet is részletekbe menően ismerni kell hozzá (nem elegendő csak a processzor utasításkészletének ismerete!). Már a kezdetekben létrejöttek a programkészítés megkönnyítése céljából az ún. magasszintű nyelvek. Segítségükkel az emberi nyelvhez, gondolkodáshoz többé-kevésbé közelebb álló módon hozhatunk létre alkalmazói programokat. Egy *fordítóprogram* (compiler) lefordítja a magasszintű utasításokat a gép számára emészthető kódokra, egy *szerkesztőprogram* (kapcsolatszerkesztő, linker) pedig összehangolja, összeilleszti a különböző időben, vagy különböző programozók által megírt programrészeket.

Operációs rendszer

Van azonban egy további megoldandó probléma is. Az elkészített, vagy vásárolt alkalmazási programokat nemcsak egy bizonyos gépen szeretnénk futtatni, hanem szeretnénk hazavinni, munkahelyünkön használni vagy akár eladni. Úgy kell tehát elkészíteni őket, hogy a számítógépek minél szélesebb körén működőképesek legyenek. A felhasználók nem minden programhoz vásárolnak új számítógépet, tehát ugyanazon a processzoron, illetve rendszeren az alkalmazások több generációjának is futnia kell. Az állítás fordítva is igaz, azaz egy bizonyos alkalmazás többféle rendszeren is működőképes kell legyen (a video vezérlők, hangkártyák és egyéb perifériák százain kell futnia), sőt a processzor konkrét típusától lehetőleg függetlenül (természetesen csak ha a „családon" belül maradunk). Szükség van tehát a szoftverek egy olyan csoportjára is, amely a felhasználói alkalmazások és a hardver közé ékelődve „eltakarja" az alkalmazások elől a hardver különbözőségét és számukra egységes „virtuális" gépként viselkedik. Ez a szoftver az *operációs rendszer*. Az alkalmazásokat tehát legtöbbször az operációs rendszerhez és nemcsak a processzorhoz készítik.

Hálózatok

A számítógépek összekapcsolása, helyi, majd világméretű hálózatba szervezése, szinte beláthatatlan lehetőségeket nyitott meg az informatika fejlődése előtt, lassan (?!) egész életünkre hatással lesznek (pl. internetes kávéfőző, mikrohullámú sütő stb.). A hálózati szoftverek tárgyalásánál nem kerülhetünk meg bizonyos hardver vo-

natkozásokat sem, de általános tendencia, hogy ez az oldal fokozatosan a specialisták kis csoportjának érdeklődési körébe szorul. Ez a terület erősen feszegeti már magának a számítástechnikának a határait is (az internetes telefon számítástechnika vagy telekommunikáció?), ezért ma sokkal közkedveltebb az informatika, illetve a telematika kifejezés.

A továbbiakban az operációs rendszerekkel kapcsolatos legfontosabb fogalmakat, feladatokat tárgyaljuk, majd megismerkedhetünk a programok összeállításának, készítésének egy megközelítésével, végül a számítógép-hálózatok világára vetünk egy pillantást.

2. Az operációs rendszer

! Az operációs rendszer egyik alapfeladata tehát az, hogy a felhasználót és a programozót megkímélje a közvetlen hardverkezelés keserveitől.

(A másik alapfeladat az erőforrások hatékony elosztása és a versenyhelyzetek kezelése, azonban ez a funkció az alkalmazások futtatása és a programkészítés szempontjából kevéssé érdekes, ezért itt nem is tárgyaljuk.)

2.1. Az operációs rendszer felépítése

Többfeladatos rendszerekben (ma már gyakorlatilag minden operációs rendszer ilyen) az operációs rendszer csak úgy tudja ellátni feladatát, ha áthatolhatatlanul beékelődik a felhasználói szoftverek és a hardver közé. Gondoljunk csak egy egyszerű példára: mi történne, ha egyszerre több futó alkalmazás reagálna az egér mozgására, vagy a programok minden koordinálás nélkül írogatnának a nyomtatóra. Tehát minden az operációs rendszeren keresztül, annak tudtával és beleegyezésével történik. *Az operációs rendszereken belül* ! *meg szokták különböztetni a rendszer magját, a kernel-t, és a felhasználók felé közvetlen kapcsolatot biztosító részt, a burkot, a shell-t.* Ez utóbbi tulajdonképpen felfogható egy olyan felhasználó alkalmazásként is, amely különösen kedves az operációs rendszer szívének.



A shell-rõl a késõbbiekben még lesz szó, vizsgáljuk meg alaposabban a kernel-t, pontosabban annak hardver, illetve szoftver felõli felületeit!

2.1.1. A rendszermag alsó (hardver) felülete

Az I/O eszközök általában megszakítás kéréssel jelzik, ha kiszolgálásra van szükségük, a processzor pedig – ha más, fontosabb dolga nem akad éppen – elindítja az operációs rendszer megfelelő rutinját. Azt, hogy éppen melyik rutin a megfelelő, a külön megalkotott megszakítás vezérlő dönti el annak alapján, hogy a kérést reprezentáló jel melyik vezetéken érkezett (PC-kben egyszerűen megszorozza a vezeték sorszámát 4-gyel, és az így kapott memóriacímről – a megszakítási vektortáblából – kiolvassa a rutin címét.) A kiszolgáló rutin az esetek többségében a hardvert kell vezérelje (például a lemez olvasófejét kell egy adott helyre irányítani). A hardver oldal azonban – mint már említettük – meglehetősen változatos képet mutat, ezért ebben a kérdésben (is) az operációs rendszer némi segítségre szorul. Mennyivel egyszerűbb lenne, ha minden periféria ugyanolyan képet mutatna! A probléma kezelésére több megoldás is született, nézzünk néhány tipikus példát!

1. **Klasszikus perifériák.** Billentyűzete, monitora például majdnem minden számítógépnek van, attól függetlenül, hogy milyen operációs rendszer található rajta. Másrészt az operációs rendszer maga is lemezről töltődik be, tehát szükség van operációs rendszertől független lemezkezelésre is. Ezen az alapvető funkciókat ellátó rutinokat az alaplapon olyan formában érdemes megvalósítani, hogy kikapcsoláskor is megmaradjon a programkód (Ezt a feladatot látja el a BIOS – Basic Input Output System – alapvető ki- és beviteli rendszer).
2. **Egységes hardver felület.** Ha a merevlemezeket, CD olvasókat belső felépítésüktől, tulajdonságaiktól függetlenül olyan elektronikával látjuk el, amely a többi egység felé már egyformán viselkedik, nagy gondot vehetünk le az operációs rendszer válláról. (IDE - Integrated Device Electronics - beépített eszközvezérlő elektronika)
3. **Eszközmeghajtók a gyártótól.** Az operációs rendszer ez esetben egy egységes, szabványos felületet mutat az I/O eszközök felé (HAL - Hardware Abstraction Layer - elvonatkoztatott hardver felület), az eszközök ehhez a felülethez történő illesztését biztosító programocskát a hardvergyártó hivatott szállítani (Device Driver - eszközmeghajtó). Olykor – például a VGA kártyák esetén – az eszközmeghajtó program magán a kártyán helyezkedik el egy ROM-ban.
4. **Intelligens perifériák.** Készítsünk olyan interfészt, amely az operációs rendszer által feltett kérdésekre értelmes válaszokat tud adni, így meg lehet vele „beszélni" a szükséges beállításokat, az eszközvezérlés módját. Nem kell tehát semmi mást tenni, mint áramkörileg csatlakoztatni az eszközt, és (a kommunikáció után) már működik is (PnP – Plug and Play – csatlakoztasd és használd).

2.1.2. A rendszermag felső (szoftver) felülete

A felhasználói alkalmazások kommunikációja a kernellel nagyon hasonlít a hardverkezelés folyamatához. Lényegében itt is megszakításkezelésről van szó, de a kérés közvetítője nem egy vezeték, hanem egy speciális gépi kódú utasítás (INT), melynek paramétere (szintén 4-gyel szorozva, azaz kettővel balra léptetve) határozza meg

a kiszolgáló rutin címét. *Ezt a szoftver eredetű megszakítást nevezzük rendszerhívásnak (system call).* A paraméterátadásra, az eredmény visszaadására, valamint az esetleges alfunkció kiválasztására a regiszterek szolgálnak. A szoftver megszakítások funkciója és sorszáma közötti kapcsolat természetesen rögzített, és az operációs rendszerek készítői – jól felfogott érdekükben – arra is ügyelnek, hogy a fejlettebb rendszerek elődeikkel kompatibilisek legyenek.

2.1.3. A rendszermag „magja" (röviden)

A rendszermag közvetít a hardver és a felhasználói szoftver között, itt történik *az egyes kiszolgálás kérések koordinálása, a rendelkezésre álló erőforrások „igazságos" és hatékony elosztása.* A közvetlen perifériakezelő funkciókon felül gazdálkodni kell a processzor idővel, a memóriával, készenléti sorokat kell fenntartani azon folyamatok számára, amelyek éppen várakozni kényszerülnek, illetve átmeneti tárolókat létesíteni az eszközök különböző sebességét kiegyenlítendő. Nagyon fontos, gyakran kiemelt feladat az állománykezelés, tehát érdemes ezt egy kicsit részletesebben is megvizsgálni.

2.2. Állománykezelés

Mivel a számítógépeket adatok, illetve programok kezelésére és tárolására használjuk, minden operációs rendszer nagy súly fektet ezek könnyű és gyors elérhetőségének biztosítására (pl. DOS – Disk Operating System, azaz lemezkezeléssel kiegészített operációs rendszer). Az *adatok állományokba (fájlokba) csoportosíthatók* így rájuk a továbbiakban egyszerűen egy logikai névvel hivatkozhatunk. Nem kell tehát a felhasználóknak azzal foglalkozniuk, hogy állományaik egy lemez melyik sávján, azon belül is melyik blokkban helyezkednek el, hanem nyugodtan rábízhatják magukat az operációs rendszerre, mely a megadott név alapján előkeresi számukra a megfelelő programot, vagy adatokat.

Állomány (fájl, file): Háttértárolón tárolt adatok együtt kezelt, névvel rendelkező összefüggő csoportja.

A fájl fogalmán általában tárolt adatokat értünk, de léteznek olyan operációs rendszerek (például a UNIX), ahol ezt a szemléletet minden külső adatfolyamra, így a képernyő tartalomra és a billentyű-

zetre is általánosították. A fájl fogalom kiszélesítése a programozó számára rendkívül kellemes, ezért egyes elemeit minden operációs rendszerben megtalálhatjuk. A programozó dolga nagyban leegyszerűsödik, mivel nem kell törődnie azzal, hogy programja milyen perifériáról kap majd bemenő adatokat, vagy eredményeit nyomtatóra, képernyőre, fájlba kell kiírnia, vagy csupán át kell adnia egy másik programnak, elegendő egy fájlra hivatkozni.

2.2.1. Elnevezések, hivatkozások

A felhasználók és programok fájlnevek segítségével hivatkoznak a kívánt adatcsoportokra. Ezeket a neveket általában a felhasználó adja, de előfordulhat az is – ideiglenes állományok esetén –, hogy az operációs rendszert kérjük fel névadásra, mely a többi elnevezés ismeretében ügyel az egyediségre.

! *A fájlneveket alkotó karakterek lehetséges halmaza, a név lehetséges hossza, valamint szerkezete az aktuális operációs rendszer függvénye.* A nevek több összetevőből állhatnak, melyeket valamilyen speciális karakter választ el egymástól. A leggyakoribb a két összetevőből álló név, ahol az *első az egyedi, a fájl tartalmára utaló rész, a második rész a fájl jellegére (szöveg, program, adatbázis) utal.* Az elnevezésekre vonatkozó szabályok rendkívül változatosak, nézzünk néhány példát!

Az **MS-DOS**, személyi számítógépeken klasszikusnak számító operációs rendszer nagyon szigorú szabályokat követel meg. A fájlnév két komponensből áll, az első rész, a *név*, minimum 1, maximum 8 karakterből állhat, az elválasztó *pont* után következő *kiterjesztés* legfeljebb 3 karakterből állhat (itt alsó határ nincs, azaz el is maradhat). A fájlneveket alkotó karakterek az ASCII kódtábla nagybetűi, számai és egyes speciális karakterei (pl. ~!@#$%^&_-{}) közül kerülhetnek ki. A DOS nem tesz különbséget kis- és nagybetűk között, az operációs rendszer minden karaktert automatikusan nagybetűvé konvertál. Az újabb DOS változatok megengedik ugyan ékezetes karakterek használatát is, de ezek használata az egyes gépek eltérő konfigurációja, valamint az említett karakter konverzió miatt hord némi veszélyeket magában.

A **UNIX** fájlnevek hossza maximum 255 karakter lehet, és tetszőleges számú, a DOS-hoz hasonlóan *ponttal* elválasztott összetevőkből állhatnak. A UNIX megkülönbözteti a kis- és nagybetűket, így a

FÁJL, fájl és Fájl három különböző állományt jelöl. A speciális, vagy ékezetes karakterek használatára nagyjából a DOS szabályai érvényesek, annak veszélyeivel együtt, azaz nem szerepelhetnek olyan karakterek, melyek az operációs rendszer számára valamilyen speciális jelentéssel bírnak.

A **Windows 95** speciális kettős elnevezésrendszert használ, részben a dokumentum-orientált szemlélet, részben a DOS kompatibilitás miatt. A Windows 95-ben minden fájlnak két neve van, egy rövid, és egy hosszú. A rövid fájlnevek követik a hagyományos DOS konvencióit, azaz 8+3 karakteres felépítésűek. Minden fájlhoz tartozik azonban egy legfeljebb 250 karakter hosszúságú, tetszőleges karakterekből álló név is, ugyancsak maximum három karakteres kiterjesztéssel. A Windows 95 alkalmazások ezt a hosszú nevet látják, ebből képezik a rövid nevet. Az első nyolc karakter automatikus választása azonban nem ad kielégítő megoldást, hiszen lehet sok olyan hosszú név is, melynek eleje megegyezik. A Windows 95 úgy oldja meg ezt a kérdést, hogy a hosszú fájlnévből eltávolítja szóköz karaktereket, majd az első 6 karakter után egy elválasztó jelet (~), azután egy sorszámot ad.

A fájlokra a felhasználói folyamatok általában a pontos nevük alapján hivatkoznak, de lehetséges a név egy részlete alapján is keresni egy, esetleg több, a mintára illeszkedő állományt. A többféle tartalommal is kitölthető, úgynevezett *helyettesítő karakterekre* (egyéb elnevezésük wildcard, joker, maszk metakarakter) példa a DOS esetén a '?' mely tetszőleges karakter egyetlen előfordulását jelenti, valamint a '*', mely tetszőleges tartalmú és hosszúságú karakterlánc helyett állhat. A UNIX a fenti lehetőségeken kívül ismeri a *karaktercsoport* lehetőségét is, melynek lehetséges értékeit szögletes zárójelek között kell megadni. Az '[ABC]' kifejezés például állhat egy 'A', egy 'B' vagy egy 'C' karakter helyett.

2.2.2. Az állományok egyéb jellemzői

A fájlokhoz a nevükön kívül egyéb információk is tartoznak, melyeket részben az operációs rendszer ad, részben a felhasználó. A többletadatok a különböző operációs rendszerekben a fájlnevekhez hasonló nagy változatosságot mutatnak.

Az **utolsó módosítás időpontja** mindig szerepel az adatok között. Újonnan alkotott, még nem módosított fájl esetén ez a paraméter természetesen megfelel a létrehozás időpontjának. A **fájl mérete** szintén nagyon fontos információ, általában bájtokban adják meg. Több felhasználós rendszerek esetén rögzítésre kerül a **fájl tulajdonosa** is.

A fájl állapotára utaló jelzőbiteket összefoglaló néven **attribútumoknak** nevezzük. (Az alábbiakban a DOS és a UNIX által használt jellemzők egyvelegét ismertetjük, melyek együtt talán képet adhatnak a lehetséges funkciókról) Az attribútumok jelezhetik az archiválást végző program számára, ha egy fájl megváltozott az utolsó mentés óta, azaz *archiválandó* (archive needed), az operációs rendszer többi folyamata számára, hogy a fájl *csak olvasható* (read only), *rendszerfájl* (system), *rejtett állomány* (hidden). A speciális tulajdonságú, adminisztrációs fájlokat is attribútumok különböztetik meg a felhasználói állományoktól, különböző jelzőbitjei vannak a *katalógusok*nak (directory) (amiket nemsokára tárgyalunk), a *szimbolikus hivatkozások*nak (link), az ideiglenes, *adatcsere fájlok*nak (pipe).

Néha (például a UNIX esetében) az felhasználók **hozzáférési jogait** is a fájlok jellemzői között találjuk, azaz itt kerül szabályozásra, hogy ki írhatja, olvashatja az állományt, vagy - programfájl esetén - hajthatja vére azt.

2.2.3. Katalógusok (directory)

A katalógus olyan speciális, adminisztratív célokat szolgáló fájl (ezért elnevezésére is ugyanazok a szabályok vonatkoznak, mint az állományokéra), amely a lemezen lévő fájlok adatainak listáját tartalmazza. Magát a katalógusfájlt a felhasználók általában közvetlenül nem láthatják, tartalmukat rendszerhívások segítségével érhetjük el.

(Az eredeti angol elnevezést, a *directory*-t magyarul gyakran *könyvtár*nak is nevezik, de tartalmilag helyesebb, és a függvény- vagy szubrutin könyvtáraktól való megkülönböztetést is jobban szolgálja a *katalógus* szó.)

Katalógus (könyvtár, directory): Olyan speciális állomány, mely az adatokat tartalmazó állományok jellemzõit tárolja, lehetõvé teszi csoportosításukat.

Ha egy folyamat egy fájlra hivatkozik, az operációs rendszer először a katalógust vizsgálja meg, hogy létezik-e egyáltalán ilyen. Pozitív válasz esetén jön a következő ellenőrzés, hogy a felhasználónak, illetve az általa indított folyamatnak van-e joga a kívánt művelethez. Amennyiben a fájl is létezik, és a jogosultságok is rendben vannak, akkor kezdődhet meg a katalógusban lévő, a fizikai elhelyezkedésre utaló információ alapján a művelet végrehajtása.

A mai operációs rendszerek szinte kivétel nélkül *hierarchikus*, vagy más néven *fa struktúrájú katalógus rendszert* használnak. A hierarchikus rendszer kiindulópontja a gyökérkönyvtár, mely tartalmazhat fájlokat és alkatalógusokat (subdirectory), ez utóbbiak szintén tartalmazhatnak fájlokat és alkatalógusokat és így tovább. Az elrendezés hasonlít egy fejre állított fára (innen az elnevezés), melynek gyökere a gyökérkönyvtár, ágai az alkatalógusok, levelei a fájlok. A hierarchikus felépítés a fájlok keresésének idejére is kedvező hatással van, mivel a lineáris rendszerekkel szemben itt alkalmazható a lényegesen gyorsabb bináris keresés.

2.2.4. Kötetek (volume)

A háttértárat az operációs rendszerek általában logikai kötetekként kezelik (pl. DOS, Windows, NetWare, de a Unix NEM!). A kötetek speciális nevet, esetleg címkét kaphatnak.

Kötet (meghajtó, volume, drive): Az a névvel ellátott logikai egység, mely segítségével az operációs rendszer a számítógép háttértárolóit kezeli.

A személyi számítógépek világában az angol ábécé 26 betűjét használják egy kettősponttal megtoldva (pl. a:, b:, c:, .. z:). Mindegyik kötethez tartozik egy gyökérkatalógus, ebben helyezkednek el a további alkatalógusok, illetve állományok adatai. *A DOS hagyományokon alapuló operációs rendszerekben az elsődleges hajlékony lemez meghajtó az A:, az elsődleges merevlemez a C: nevet viseli, a hálózati meghajtók az F: .. Z: tartományt kapták.*

Meg kell jegyezni, hogy hálózatok használata esetén a 26 betű kevésnek bizonyulhat, ilyenkor a kötetnév általában több karakterből áll, pl. SYS:, USER:.

2.2.5. Hivatkozások

Ha egy állományra hivatkozni akarunk, nemcsak a nevét kell megadni, hanem azt a kötetet, illetve katalógust is, amelyben megtalálható.

! Az **abszolút hivatkozás** esetén a kötet megadása után a gyökér katalógustól kezdődően az összes közbülső katalógus nevének felsorolása után jutunk el a fájlhoz

! Az **aktuális kötet / katalógus** (current drive / directory) fogalmának bevezetésével egyszerűsíthető a hivatkozás, az eddigi, a gyökérkatalógustól induló *abszolút címzéssel* szemben alkalmazható az aktuális helyzethez viszonyított *relatív címzés*. Az aktuális kötet, illetve (kötetenként) az aktuális katalógus nevét az operációs rendszer saját adatterületén tárolja. Ez gyakran a képernyőn is megjelenik (pl. DOS prompt) vagy egyszerű paranccsal kiíratható (pl. Unix cwd).

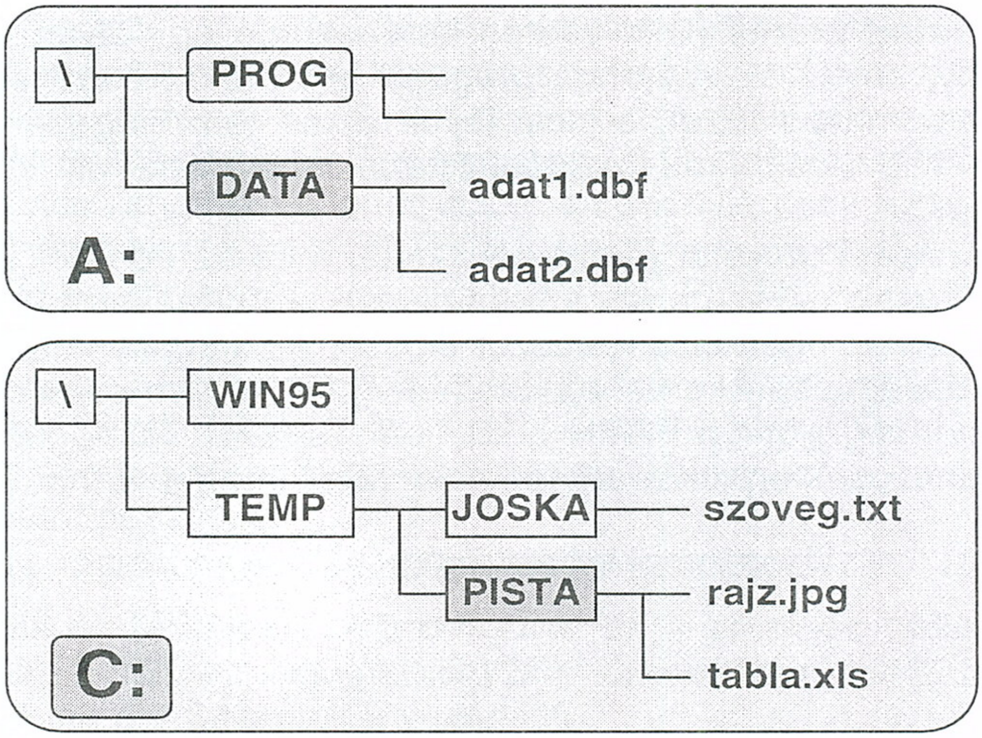
! A **relatív hivatkozás** tehát a fájl megadásának az a módszere, amelyben a gyökér katalógus helyett az aktuális katalógus a kiinduló pont.

A gyökérkatalógus neve általában ‚\' (backslash), azaz egy fordított törtvonal.

Fa struktúrát alkalmazó rendszerekben az egyes katalógusoknak lehetnek szülei és gyermekei. A szülőkkel a szorosabb kapcsolat természetes, ezért igazi nevén felül, már csak szülői mivoltából is indokolt külön névvel illetni. A szülő katalógus neve általában (a gyermek szemszögéből) '..' (azaz két darab pont), az aktuális katalógus relatív neve '.' (azaz egy darab pont).

Az abszolút és relatív hivatkozás között a legszembetűnőbb formai különbség, hogy az abszolút hivatkozás (a meghajtó neve után) mindig tartalmaz ‚\' karaktert.

Nézzünk egy példát!



V.2. sz. ábra
Példa fa struktúrájú katalógusrendszerre

A példában két kötetünk van, az aktuális meghajtó a `C:`, az aktuális katalógus az `A:` illetve `C:` meghajtón rendre a `\DATA`, illetve a `\TEMP\PISTA` (az ábrán sötétebb árnyalattal jelölve).

Abszolút címzések:

```
A:\DATA\ADAT1.DBF
C:\TEMP\PISTA\RAJZ.JPG
C:\TEMP\JOSKA\SZOVEG.TXT
```

Relatív címzések:

`A:ADAT1.DBF`	(aktuális katalógus)
`C:RAJZ.JPG`	(aktuális katalógus)
`\TEMP\JOSKA\SZOVEG.TXT`	(aktuális kötet)
`..\TEMP\JOSKA\SZOVEG.TXT`	(aktuális kötet, katalógus)

2.2.6. Közvetett elérés, keresési útvonal

A gyakran használt, de a logikai fájlrendszerben különböző helyeken található programok könnyed indítására a relatív hivatkozás lehetősége sem nyújt elfogadható megoldást. Legtöbb operációs rendszernél alkalmazható a közvetett elérés, vagy a keresési útvonal technikája.

Közvetett elérés (link) esetén létrehozható egy olyan kis méretű állomány, amely az indítandó program futtatásához szükséges adatokat tartalmazza. Ezek a kicsi fájlok aztán tetszőlegesen elhelyezhetők, csoportosíthatók (például kitehetők a Windows 98 „munkaasztalára").

A **keresési útvonal** (path) nem más, mint az operációs rendszer egy „változója". Egy parancs begépelése, vagy egyes fájl megnyitások esetén az operációs rendszer először az aktuális köteten, az aktuális katalógusban keresi az állományt, de kudarc esetén sem esik kétségbe, végigjárja a PATH változóban felsorolt katalógusokat, és a hibaüzenet csak ennek eredménytelensége esetén várható.

Pl.

```
PATH C:\WINDOWS;C:\NORTON;C:\WINDOWS\COMMAND
```

2.2.7. Fájlok fizikai elhelyezése

A számítógépek kiemelkedő fontosságú háttértára a merevlemez (hard disk drive, winchester), ezért a fájlok elhelyezését ezen a példán mutatjuk be, ennek is azt a változatát, amelyet az IBM PC-k terjesztettek el. Az egyszerűség kedvéért még azt is feltételezzük, hogy csak egyetlen merevlemez található a képzeletbeli számítógépünkben.

A merevlemezeken a tárolható legkisebb kezelhető egységek a blokkok, azonban ezek gyakran értelmetlenül kicsinek bizonyulnak, így az operációs rendszer számára a legkisebb egység a több blokkból álló fürt, az úgynevezett **cluster**. A lemezeket ezentúl úgy kezelhetjük, mintha egy adott számú cluster-ből álló tárolósorozat lenne (a fejek mozgatását a lemezkorongok között bízzuk az operációs rendszerre).

Az operációs rendszer elhelyezkedése, betöltése

A merevlemezek részekre, úgynevezett **partíciókra** oszthatók. (A partíciók maximális száma 4.) A felosztásra vonatkozó adatok, azaz az egyes partíciók kezdő, illetve utolsó clusterének sorszáma a lemez legelején lévő táblázatban, az úgynevezett **partíciós táblában** találhatók. Ugyanitt van az a kis programocska is, a **Master Boot Record** (MBR), amely a táblázat adatait értelmezni tudja, és a megfelelő partícióról elindítja az operációs rendszer betöltését. (Ha az

operációs rendszer fogalmát tágabb értelemben használjuk, helyesebben azt kellene mondani, hogy az operációs rendszer *maradékának* betöltését, hiszen a BIOS már rendelkezésre áll!) De a sok közül melyik a *megfelelő* partíció? Egy lemezen – különböző partíciókban – több operációs rendszer is lehet, azonban közülük csak az egyik aktív. Az aktivitást szintén a partíciós tábla egy mezője tárolja, amelyet az operációs rendszer segédprogramjával (pl. FDISK), vagy különböző betöltést felügyelő (boot menedzser) programokkal lehet megváltoztatni. Minden partíció elején megtalálhatjuk a **Boot Record**-ot, és a hozzá tartozó adattáblákat, melyek azonban – az MBR-rel ellentétben – már operációs rendszer specifikusak, és a konkrét operációs rendszer betöltését végzik.

Tehát az operációs rendszer betöltésének lépései:

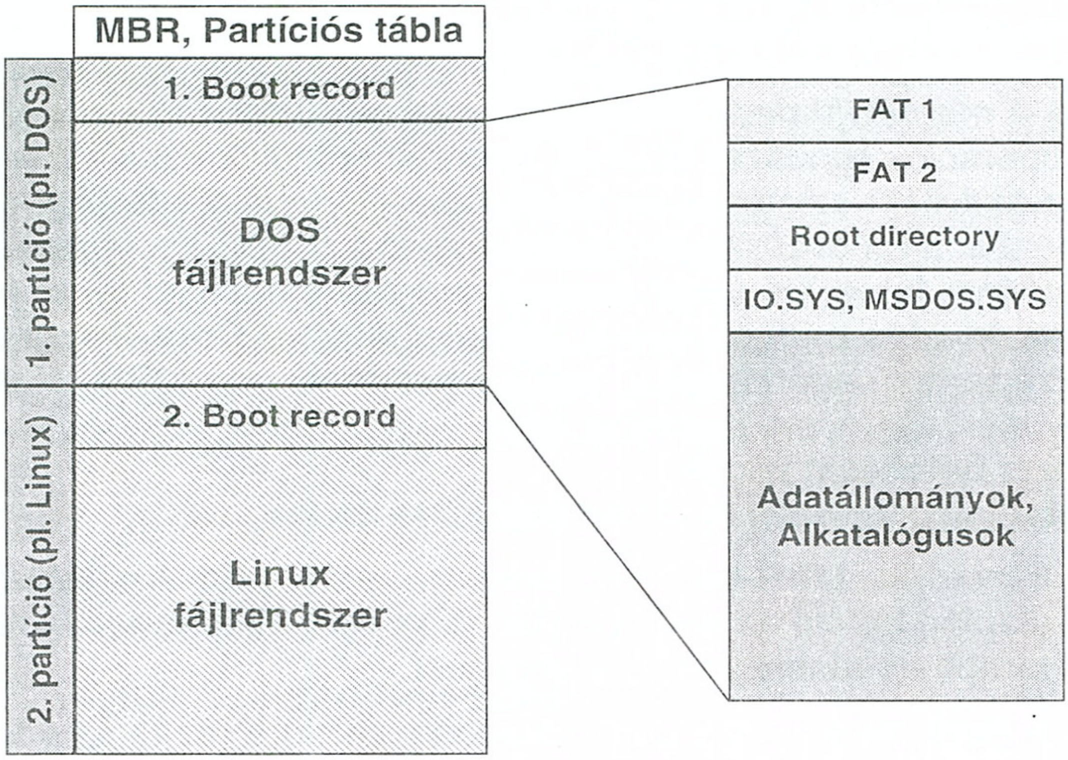
1. Elindul az MBR-ben lévő program, és a partíciós tábla adatai alapján betölti az aktív partíció Boot rekordjában található programot, és átadja annak a vezérlést.
2. A Boot rekord programja az adott partícióra (azaz az adott operációs rendszerre) jellemző paraméterek segítségével megkeresi az operációs rendszer első állományát (pl. IO.SYS) és továbbadja a vezérlést.
3. A többi már az operációs rendszer dolga ...

Az operációs rendszer állományai (a partíció határokhoz képest) állandó helyen találhatók, azonban a felhasználói állományok esetén ez a módszer természetesen nem alkalmazható.

Felhasználói állományok elhelyezése

Minden partíció (ritka kivétellel) egy logikai kötetként jelenik meg. A továbbiakban egy partícióra szorítkozunk, ezen belül vizsgáljuk a fájlok elhelyezését.

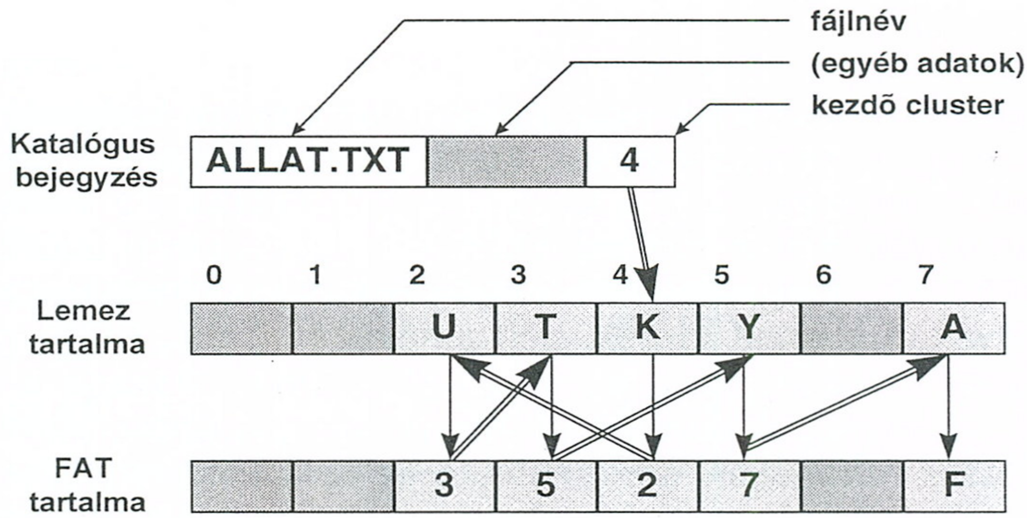
Ahhoz, hogy adatokat, programokat tárolhassunk a köteten, azt elő kell készíteni, létre kell hozni rajta az adminisztrációhoz feltétlenül szükséges táblázatokat. Ez az eljárás a lemez **formázása**, melynek során felmásolódhat az operációs rendszer magja, lefoglalódik a hely a gyökérkatalógusnak (root directory), amely a fájlrendszer kiinduló pontja lesz, és létrejön (biztos, ami biztos két példányban) a fájlok elhelyezését regisztráló táblázat, a **FAT** (File Allocation Table). Nézzük, hogy néz ki egy partíció!



V.3. sz. ábra
Állományok elhelyezése a lemezen

Az állományok fizikai elhelyezésére több módszer létezik, de az alábbiakban csupán a PC-s világban leginkább elterjedt láncolt elhelyezést mutatjuk be. A katalógusban csak a fájl kezdő clusterét kell megadni, az összes többi adatot a fájl elhelyezési tábla (FAT) tartalmazza. A táblázatnak ugyanannyi eleme van, mint ahány cluster a lemezen és minden rekesz tartalma a fájl következő clusterére mutató sorszám, ha van ilyen, csupa egyes (azaz 16 bites FAT esetén FFFF), ha ez volt az utolsó blokk. Ezzel elérhető, hogy az állományoknak nem feltétlenül kell folytonosan elhelyezkedniük a lemezen, mivel az egymást követő clusterek sorszáma a FAT alapján megállapítható.

Legyen példánkban egy cluster mérete mindössze 1 bájt, és legyen csupán 8 clusterünk. A memóriában lévő 5 bájtot (`KUTYA`) `ALLAT.TXT` néven szeretnénk lemezre menteni. (Ebben az esetben ez a művelet az operációs rendszernek nem túl jól sikerült, de a nyilakat figyelmesen követve felfedezhetjük az eredeti szót.)



V.4. sz. ábra
Állományok láncolt elhelyezése a lemezen

A FAT meglehetősen nagy táblázat lehet, és szerepe döntő. Ezen kívül nagyon sűrűn kell használni, ezért a memóriában kell tartani, ami még szűkösebb, mint a háttértár. A FAT sérülése esetén nagy valószínűséggel a kettészakadt fájlt visszaállítani nem lehet. A láncolási módszert alkalmazó operációs rendszerek, például a DOS, a Windows és a NetWare a biztonság kedvéért két ilyen táblázatot tartanak fenn.

2.3. A felhasználói felület

A legelső dolog, amivel a felhasználó találkozik, az operációs rendszerek felhasználói felülete.

Az eddigiekben megmutattuk, hogy az operációs rendszerek szinte minden mozzanatot felügyelnek. Kiszolgálják, ütemezik a folyamatokat, biztosítják számukra a megfelelő erőforrásokat. A mai rendszerek túlnyomó többsége interaktív, tehát a felhasználó is igényli, hogy befolyásolhassa a rendszer működését. Miért ne biztosíthatnák ezeket a szolgáltatásokat nemcsak a felhasználói folyamat, hanem közvetlenül a felhasználók számára is?

Van még ezen kívül egy egyáltalán nem elhanyagolható szempont. Lehet, hogy egy operációs rendszer sokat tud, de egyet biztosan nem. Nem tudja kitalálni, hogy egy felhasználó mit akar, milyen programot szeretne indítani. A felhasználói felületre tehát feltétlenül szükség van.

2.3.1. Karakteres felhasználói felület, a parancsértelmező

A parancsértelmezők az interaktív rendszerek kialakulásával jelentek meg, a parancsnyelvek továbbfejlesztéseként. Feladatuk az operációs rendszer szolgáltatásainak biztosítása az interaktív felhasználó számára. A funkció többféle elnevezésével találkozhatunk attól függően, hogy a feladatkör mely részét tekinthetjük elsődlegesnek. Gyakori a **shell** (burok, héj) elnevezés (Unix, DOS) mely arra utal, hogy a felhasználói felület burokba zárja, eltakarja a rendszer magját, a kernelt. A **command interpreter** (parancsértelmező) kifejezést azok használják, akik számára az operációs rendszerek fő feladata az adott parancsok kiszolgálása. Olykor találkozhatunk még a *monitor* (felügyelő) névvel, ha a folyamatok kezelése, nyomon követése az elsődleges. A grafikus rendszerek (pl. a Windows-család tagjai) a feladatkört több részre osztják, így létezik program, fájl-, nyomtató-, feladat (task) kezelő moduljuk.

A továbbiakban a megjelenés formáitól eltekintünk, a funkció megjelölésére a „shell" nevet használjuk. A shell-ek, mint már említettük, tulajdonképpen olyan felhasználói programok, amelyek az operációs rendszer lelkéhez közel állnak. Működésük nem igényel kivételes, privilegizált módot. Bizonyos esetekben - ha egy felhasználó mindig kizárólag ugyanazt a programot futtatja - a shell el is maradhat, a felhasználó számára a felületet maga a felhasználói program jelenti.

Más esetben az egyes felhasználók, vagy alkalmazási módok más-más felhasználói felületet kedvelnek. A többféle felület a UNIX-nál természetes (C, Bourne-shell), de ha egy kicsit tágabb értelmezést is megengedünk, shell-nek tekinthető a DOS esetén a Norton Commander, a PCShell vagy a DOS-Shell, Windows esetén a Fájlkezelő és a Programkezelő is. A shell alapvető feladatai tehát:

- programindítás, programkezelés
- egyéb, operációs rendszer funkciók felhasználói szintű biztosítása (általános értelemben vett fájlkezelés)

A DOS parancsok általános felépítése a következő (a szögletes zárójel azt jelenti, hogy az így jelölt elem olykor elmaradhat):

```
parancs [paraméterek] [/kapcsolók]
```

A parancs határozza meg az elvégzendő feladatot, a paraméterek mutatják, hogy azt mely objektumokon kell végrehajtani, a kapcsolók (switch) kissé módosítják a parancs mûködését.

A legfontosabb DOS parancsok:

`programnev`	a `PROGRAMNEV` nevű program indítása
`FDISK`	a partíciós tábla módosítása
`FORMAT drive:`	kötet formázása
`drive:`	az aktuális meghajtó megadása
`MD directory`	alkatalógus létrehozása
`CD directory`	az aktuális katalógus megadása, váltása
`RD directory`	(üres) katalógus törlése
`DIR`	aktuális katalógus tartalmának listázása
`COPY honnan hova`	állomány másolása
`REN mit mire`	állomány átnevezése
`DEL file`	állomány törlése
`TIME`	idő kiírása/beállítása
`DATE`	dátum kiírása/beállítása

A fenti parancsok ismerete általában elegendő. Ha valakinek mégis hiányérzete támad, a `HELP` parancs, vagy a parancs kiadása a `/?` kapcsolóval sokat segíthet.

2.3.2. Grafikus felhasználói felületek

A felhasználó és a számítógép párbeszéde kezdetben a lyukkártyákra rögzített, majd a konzolírógépen begépelt parancsok segítségével történt, a közeli jövőben talán a kimondott szavak, majd a gondolatok irányíthatják gépeink működését, azonban jelenleg a leginkább elterjedtek, a legnépszerűbbek a *grafikus felhasználó felületek*. Az IBM kompatíbilis gépeken a Microsoft Windows különböző variációival találkozunk, a Apple a MacOS-t használja, a Unix, és általában a nagygépes operációs rendszerek az X-Window rendszert valósították meg,

A karakteres világ parancsértelmezője nem sokban különbözik egy egyszerű felhasználói programtól. Egy-feladatos rendszerekben (single task, pl. DOS) a parancsértelmező fut, amikor éppen semmi más nem történik, háttérbe vonul, amikor egy felhasználói program kap vezérlést, és annak végeztével újra előbukkan, és várja a következő parancsot.

Többfeladatos, karakteres felület-használó rendszerekben (multitask, pl. Unix) ugyan futhatnak folyamatok a háttérben, azonban ezek működését csak meglehetősen nehézkesen lehet nyomon követni, illetve befolyásolni. Milyen jó lenne, ha minden folyamatra nyithatnánk egy ablakot, amelyen keresztül megfigyelhetjük a történéseket, beavatkozhatunk, vagy ha úgy tartja kedvünk, az ablakot be is csukhatjuk (ez persze nem jelent azt, hogy a folyamat is leáll)!

Az ablakozó technikák tehát a többfeladatos rendszerekhez kötődnek, fejlődésük kezdete a 80-as évek elejére tehető.

2.3.3. Segédprogramok, alrendszerek

Már a programfejlesztői támogatás esetén is nehéz eldönteni, hogy hol húzzuk meg az operációs rendszer határait, még inkább így van ez a felhasználó által közvetlenül használt programok esetén.

A felhasználói programok leggyakrabban használt példája az interaktív rendszerekben használt tolmács nyelv, a parancs-interpreter. A parancsnyelvet gyakran nevezik **shell**-nek (héj, burok), mivel a felhasználó csak ezen keresztül érintkezhet a maggal, a kernellel. A parancsnyelv nem más, mint a rendszerhívások, illetve rendszerhívások sorozatának kiadására szolgáló magas szintű eszköz, ezért vitathatatlanul az operációs rendszerhez tartozik, annak ellenére, hogy azonos funkciók ellátására más shellek is írhatók.

A gyakran használt operációs rendszer szintű feladatok megoldására szolgáló **segédprogramok** (lemezformázás stb.) szintén közel állnak az operációs rendszerhez.

Gyakori, hogy egy speciális felhasználói igényt nem az operációs rendszer módosításával elégítenek ki, hanem egy a segédprogramok körét kiegészítő, úgynevezett **alrendszerrel** bővítik az operációs rendszer magaszintű szolgáltatásait. A programfejlesztési támogatás is felfogható alrendszerként, de egyre terjednek a grafikus és adatbázis alrendszerek is. Az alrendszerek alkalmazásának előnye, hogy nem kell módosítani hozzá az operációs rendszert, hátrányuk, hogy egy újabb szint ékelődik a felhasználó és az operációs rendszer közé. Az első interaktív rendszerek is ilyen alrendszerként kerültek megvalósításra.

A segédprogramok, alrendszerek leggyakoribb működési területei a következők:

Állományok kezelése. Az állományok másolása, áthelyezése, átnevezése, törlése, valamint a katalógusok létrehozása mellett, a segédprogramok között megtalálhatók a lemezek partícionálására, formázására szolgálók is. Tipikus segédprogram a Norton Commander, mely a leggyakrabban használt 10 fájl-művelet végrehajtását teszi hallatlanul kényelmessé.

Programfejlesztés. Szinte valamennyi operációs rendszer tartalmaz egyszerű szövegfájlok előállítására alkalmas programot (editor), valamint szerkesztőt (linker). A Unix alapú rendszereknek gyakran szerves része a C fordító is.

Adatbáziskezelés. A számítógépes rendszerek jó része adatbázisokat kezel. Ennek támogatására olykor az operációs rendszerek külön eszközöket is tartalmaznak, néha, pl. az OS400 esetén szinte elválaszthatatlanul a többi funkciótól.

Kommunikáció. A hálózatok kezelése napjainkban egyre fontosabb. Az alapvető hálózati rétegeket megvalósító folyamatok ma már szinte kivétel nélkül az operációs rendszer szerves részét alkotják.

A segédprogramok, alrendszerek és felhasználói programok lényegében nem különböznek egymástól, gyakorlatilag csak azok eredete, súlya vagy funkciója dönti el, melyik programot vagy program csoportot melyik kategóriába soroljuk.

3. Programkészítés

A operációs rendszerek egyik feladata – mint említettük – a hardver elrejtése a felhasználók elől. A rendszermag a felhasználói folyamatok oldaláról úgy látszik, mintha egy olyan számítógép (virtuális gép) lenne, amely speciális, jellemző erőforrás kezelő utasításokkal rendelkezik. A felhasználói folyamatok írói, a programozók a hardver kezelést csak kernel szolgáltatásainak igénybevételével végezhetik Ilyen feltételek mellett természetes, hogy az operációs rendszer készítőinek, a rendszer legjobb ismerőinek az rendszermag funkcióinak megvalósításán kívül támogatniuk kell a felhasználói programok készítését is. A rendszerhívások jól definiált rendszeréhez az assembly programozók közvetlenül, a magasabbszintű nyelvek kedvelői összetettebb eljárásokon keresztül férhetnek hozzá.

3.1. A forráskód elkészítése

A **forráskód** elkészítésére szinte minden rendszer biztosít egy **szövegszerkesztőt** (editor), amely a konzolról begépelt szöveget egy állományba menti. A programozási nyelv megválasztása erősen függ az elvégzendő feladattól.

A programozási nyelveket gyakran csoportosítják annak alapján, hogy a hardverhez milyen kapcsolat fűzi őket. Ebből a szempontból léteznek **hardver orientált nyelvek** (tipikus példája a Assembly), valamint **probléma-orientált nyelvek** (például Clipper, VBA stb.) Az előbbi csoport nyelveiben a programozás nem nagyon egyszerű, de az így született program a lehető leghatékonyabban képes a hardver lehetőségeit kihasználni (persze csak, ha a programozó is képes rá!). A probléma-orientált nyelvek fejlesztésénél arra törekedtek, hogy az adott feladatcsoport logikáját a lehető legjobban megvalósítsák, így a programozást megkönnyítsék, azonban ez a hatékonyság romlásával járhat. A két csoport között helyezkednek el az **általános célú nyelvek** (pl. C, Pascal), melyek segítségével a feladatok széles spektrumát lehet megoldani.

3.2. Fordítás

A forráskód alapján azt a gépi kódú programot, amely az adott processzor által ismert utasításokat, valamint a rendszerhívásokat megvalósító szoftver megszakításokat tartalmazza a **fordítóprogram** (compiler) készíti el. Az így keletkező **tárgykódú** (object - OBJ) modul az ugrások, változók abszolút címei helyére általában még a modul elejéhez viszonyított relatív címet helyettesít, úgy mintha feltételezné, hogy a program modul a 0 címtől kezdődne. Az abszolút címek már csak azért sem szerepelhetnek a tárgykódban, mert egy program rendszerint több, külön fordított modulból áll. Az egyes programrészleteket természetesen készítheti a felhasználó, de az operációs rendszer is tartalmazhat előre elkészített, **rendszerkönyvtárakba** (library - LIB) rendezett, elsősorban perifériakezelő, vagy a felhasználói felület kialakítását segítő rutinokat. A modern operációs rendszereknek csak a leginkább hardver közeli részei íródnak gépi kódban, nagy részük a kifejezetten e célra kifejlesztett C nyelven készül, így, ha más nem is, de a C fordító az operációs rendszer születésével egyidőben már rendelkezésre áll.

E fejezetben nem foglalkozunk az interpreterekkel, azaz a soronként fordító és végrehajtó magasszintű nyelvekkel.

A kernel programozói szempontból egy függvény, vagy eljárás könyvtárnak tekinthető, a programok készítéséhez szükséges információ leírásának és segédprogramoknak összefoglaló neve a legtöbb esetben API (Application Programming Interface).

Az alkalmazások írói számára biztosított rutinok általában erősen függnek a szolgáltatások szintjétől. A DOS kb. 100 eljárása alapvető operációs rendszer funkciók elérését teszi lehetővé, míg a Windows több mint 1000 függvénye a felhasználók által közvetlenül megtapasztalt felület kialakítását is támogatja. A grafikus felületek egyedülálló népszerűségének éppen ez az egyik fő oka: a programozók - többek között saját jól felfogott érdekükben - az operációs rendszer által kínált magas szintű lehetőségeket használják fel, így mind a látvány, mind a működési mód hasonló. Ez a programozási stílus a végfelhasználó számára is nagyon előnyös. Elég egy programot megtanulni, az összes többi szinte magától értetődő, a lényegtől nem vonják el a figyelmet a technikai részletek.

3.3. Szerkesztés

A **szerkesztő** (linker) feladata a tárgykódú modulok címeinek összehangolása, kereszthivatkozások feloldása, a **betölthető program** (executable - EXE) előállítása. (A szerkesztőt gyakran *kapcsolatszerkesztő*nek nevezik, azért, hogy véletlenül se lehessen összekeverni a *szövegszerkesztő*vel.) A szerkesztett program még mindig nem tartalmazhat konkrét memória címeket. Mivel a rendszerben egyszerre több folyamat fut, előre nem tudható, hogy az elkészített program a memória mely tartományára kerül. Az operációs rendszer dolgát mindenesetre célszerű megkönnyíteni úgy, hogy amennyiben a processzor lehetővé teszi, a címeket egy, a program kezdetén beállítandó alaphoz (bázis) képest adjuk meg.

A programkészítés fent vázolt menetét a korszerű integrált fejlesztőrendszerek észrevehetetlenné teszik. Látszólag egy lépésben végzik, sőt program belövési támogatással (lépésenkénti végrehajtás, változók, memóriatartalom kiírása) egészítik ki.

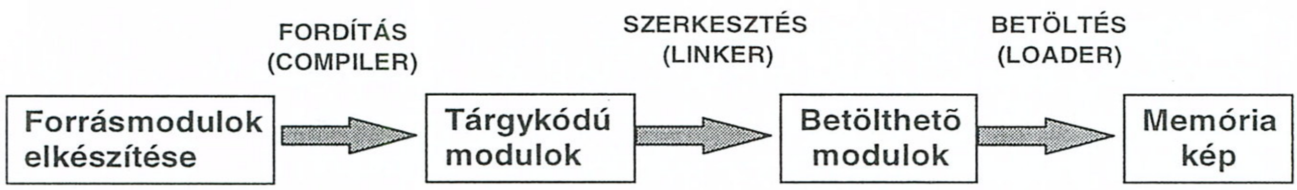
3.4. Betöltés, dinamikus könyvtárak

A **betöltő** (loader) program már az operációs rendszer magjához tartozik, feladata a végrehajtható program elhelyezése a memóriában, a bázis cím kitöltése a megfelelő értékkel, a folyamatleíró blokk elkészítése. A betöltéssel válik egy program folyamattá.

Az így elkészített, illetve betöltött program hátránya, hogy pazarló módon minden részlete egyszerre kerül a memóriába, akkor is, ha ez nem szükséges. További hátrány, hogy azok a modulok, amelyeket több program is használ, mint például a rendszerkönyvtár elemei annyiszor kerülnek betöltésre, ahány program használni kívánja őket. Az eddigi statikus szemlélettel szemben a problémát a **dinamikusan szerkeszthető könyvtárak** (Dynamic Link Library - DLL) segítségével lehet megoldani. A dinamikus könyvtárak csak akkor kerülnek a memóriába, ha azokra hivatkozás történik, bekerülésük után viszont több program is használhatja egyszerre őket.

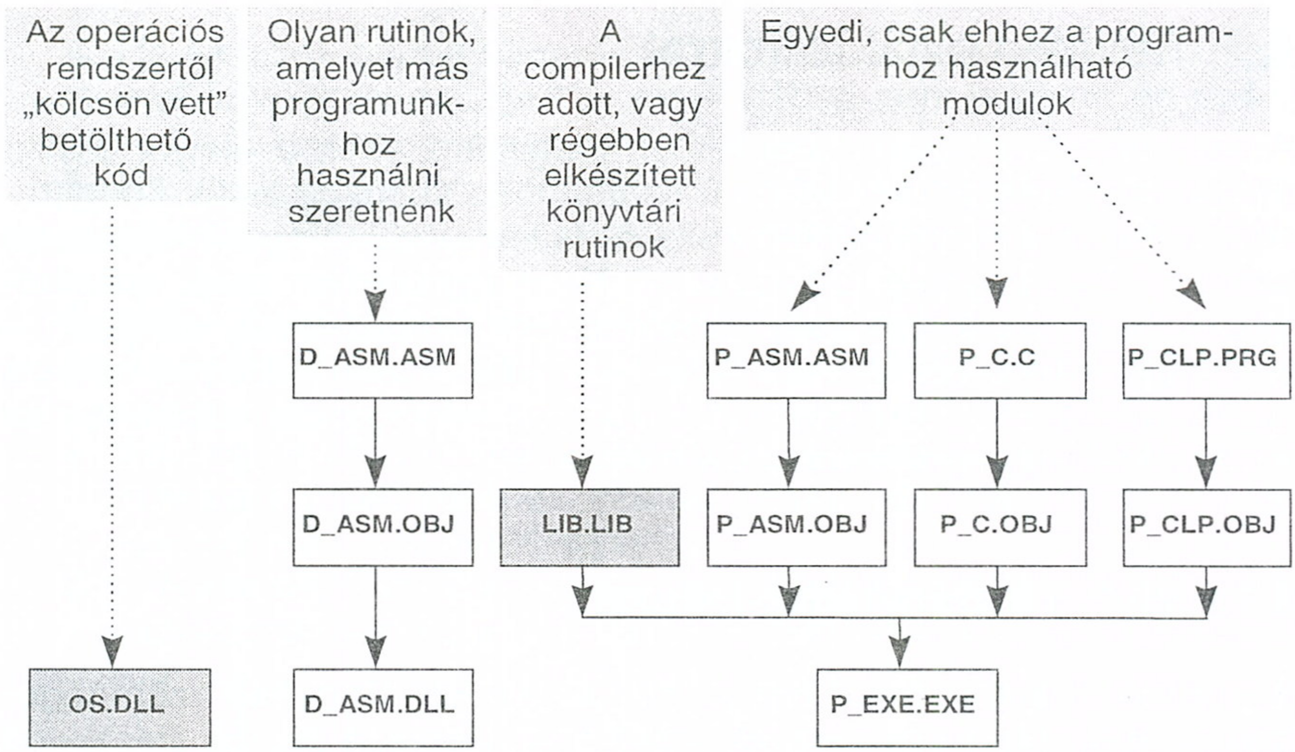
3.4.1. A program összeállítás teljes folyamata

A forráskód megírásától a memóriába való betöltődésig tartó folyamat tehát a következő.



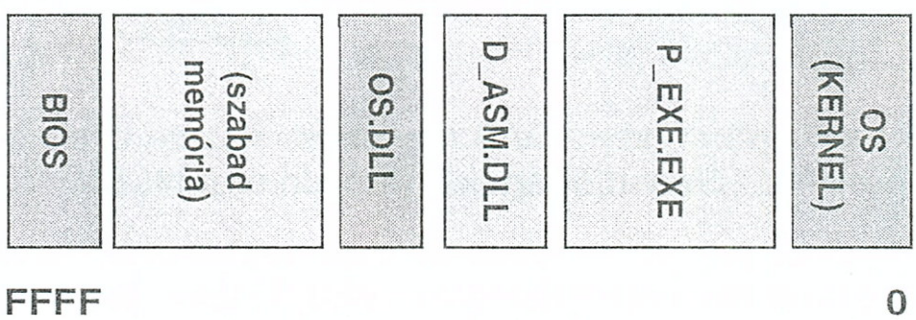
V.5. sz. ábra
A programkészítés folyamata

Nézzünk végül egy példát! Az alábbi feladat lehetne például egy beléptető rendszer programja, amelyhez felhasználjuk az operációs rendszer kernelje és a BIOS által nyújtott szolgáltatásokat (rendszerhívások), valamint az OS.DLL szubrutinjait, függvényeit. Az elkészítendő program tartalmaz adatbázis funkciókat (erre megfelelő a Clipper), hardverkezelő rutinokat (ezek Assembly-ben készülnek) a főprogram pedig C-ben készül. Az Assembly rutinok egy részét más, egyidejűleg futó programjainkhoz is szeretnénk felhasználni, ezért ezeket dinamikusan betölthető formában hozzuk létre.



V.6. sz. ábra
Egy komplex program összeállítása

A betöltődés után kialakult memória kép a következő lesz (természetesen a DLL-ek csak szükség esetén töltődnek be).



V.7. sz. ábra
Egy program részeinek elhelyezkedése a memóriában

4. Röviden a hálózatokról

A számítógép-hálózatok alapvetően megváltoztatták a világot. Eddig minden törekvés az ember és a számítógép kapcsolatának finomítására irányult, a számítógép-hálózatok kialakulásával egyre fontosabb szerepet kap a gép-gép kapcsolat. Nem egyszerűen csak számítógépek halmazáról van szó, hanem gépek *rendszeréről*, melyben – az emberi szervezethez hasonlóan – egyes részek bizonyos célokra specializálódnak. Általános trend, hogy az intelligencia súlypontja az egyedi számítógépről áttolódik a hálózatra.

4.1. Mire jó a számítógép-hálózat?

Képzeljük el a következő helyzetet! Legyen két független számítógépünk, amelyeken ugyanaz az adatbázis, illetve a hozzá illeszkedő adatbázis kezelő található (két független példányban), és mindkét géphez tartozik nyomtató is. Ilyen lehet például egy bolt két pénztárgépe. Az adatbázis ebben az esetben az árukészlet, illetve az eladások adatait tartalmazza.



V.8. sz. ábra
Számítógépek – kapcsolat nélkül

Nézzük meg, hogy a fenti rendszernek – a számítógép használatából adódó természetes előnyök mellett – milyen hátrányai vannak!

1. Ha az egyik nyomtató elromlik, az érintett konfiguráció használhatatlan, hiszen nem tud számlát előállítani
2. Mindkét számítógépnek nagy háttértárra van szüksége, hiszen az egész adatbázist külön-külön is tárolniuk kell.
3. Záráskor, vagy ha lehet gyakrabban szinkronizálni kell az adatokat, azaz biztosítani kell, hogy mindkét gép ugyanazon adatbázis alapján dolgozzon.

4. Ha az egyik gépen megsérül az adatbázis, ki kell deríteni, hogy melyik változat az „igazi" – ez általában nem is olyan egyszerű –, és át kell azt másolni a másikra.
5. Az egyik nyomtató – feltételezve, hogy nincs mindig csúcsforgalom – általában csak várakozik, nem végez hasznos munkát.

Foglaljuk össze a problémákat kissé általánosabban!
1. Az erőforrások (nyomtató, háttértár, adatbázis) kihasználtsága rossz.
2. Egy-egy erőforrás kiesése esetén – szerencsétlen esetben – a rendszer működőképes marad ugyan, de meglehetősen leromlott (50%-os) hatékonysággal.
3. Erőfeszítéseket kell tenni az adatok megbízhatóságának (szinkronitásának, integritásának) biztosítására, ez munkaerőt köt le.

Vagy még általánosabban, azért, hogy rendszerünket hatékonyabbá, és egyben megbízhatóbbá tegyük:
- **csökkenteni kell a redundanciát** (az adatbázis elegendő csak egyetlen példányban, az esetek nagy többségében elég egyetlen nyomtató is) annak érdekében, hogy rendszerünk hatékonyabb legyen; !
- **növelni kell a redundanciát** (tartalék nyomtató, az adatbázis biztonsági másolata stb.) annak érdekében, hogy rendszerünk megbízhatóbb legyen. !

(Redundanciának nevezzük egy rendszerben (vagy közleményben) azt a többletet, amely új szolgáltatást (vagy információt) nem biztosít.)

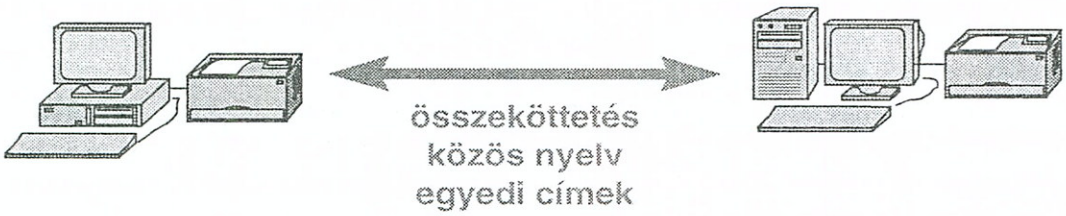
De hiszen a fenti példában van több nyomtatónk, van több adatbázis másolatunk, hogy lehetséges, hogy a rendszer mégsem megbízható?

Egy valami még hiányzik: **a kommunikáció, a kapcsolat**, mely a független rendszerelemek halmazából működő rendszer alkot.

4.2. A számítógépek közötti kommunikáció

A kapcsolathoz tehát kell (legalább) egy közvetlen vagy közvetett fizikai **összeköttetés** és (legalább) egy **közös nyelv**, amit mindkét számítógép ért. Ha pedig csak egy kicsit tovább bővítjük a kört, és 38

további számítógépeket kapcsolunk a rendszerhez, szükségünk lesz egyedi **címek**re is, melyek egyértelműen azonosítják az üzenetek címzettjét vagy címzettjeit, illetve a feladót.



V.9. sz. ábra
A gép–gép kommunikáció feltételei

4.2.1. Az összeköttetés

A kapcsolat szempontjából (mostani szempontjaink szerint) lényegtelen, hogy az milyen fizikai csatornán valósul meg. Használhatunk rézvezetéken átvitt elektromos impulzus sorozatot (koaxiális kábel, csavart érpár), valamilyen módon modulált rádióhullámokat (kábel TV, V-SAT), vagy szaggatott fénysugarat (üvegszál, lézer), lényeg az, hogy a megvalósított csatornán biteket kell átvinni szépen egymás után sorban. Ha mindegyik számítógépet mindegyikkel összekötnénk egy-egy csatornával nagyon rossz kihasználtságot érhetnénk el (és lépni sem lehetne a sok vezetéktől), hiszen egy gép általában csak egy másikkal „beszélget" és ez a beszélgetés is nagyon szakaszos, egyetlen aktív vezetéken az idő legnagyobb részében semmi sem történne.

Megoldás lehet, ha mindegyik számítógép ugyanazt a csatornát használná, és a kommunikációhoz használt üzenetcsomag önmaga tartalmazná a címzettre (és persze a feladóra) vonatkozó információt.

Gyakran nincs is más megoldás. Az 1970-es években A Xerox, a DEC és az Intel kapott olyan megbízást, hogy a Hawaii-szigetek több száz kikötője között megbízható adatkapcsolatot létesítsen. Vezetékek szóba sem jöhettek, a fénysugár alkalmazása a nagyobb távolságok miatt lehetetlen volt, egyedül az „éter"-ben terjedő rádióhullámok maradtak. A létrehozott rendszer ezért elnevezték *Ethernet*-nek.

Az adatok tehát egy csomag részei, melynek fejléce tartalmazza a címzett és a feladó címét. Ezt a technikát nevezzük **csomagkapcsolásnak** (szemben a hagyományos telefon esetén alkalmazott *vonalkapcsolás*sal). Az üzenetet tehát a hálózat minden résztvevője veszi, de csak az értelmezi, aki a saját címét a megfelelő mezőben felismeri. Íme egy tipikus csomag (keret, frame):

Csomag típusa	Adó címe	Vevő címe	Hasznos adatok	Ellenőrző összeg

V.10. sz. ábra: Tipikus csomag felépítése

Meg kell oldani azonban azt a feladatot is, hogy ezen a közös csatornán egyszerre mindig csak egy kapcsolat legyen aktív, azaz ne „beszéljenek" egyszerre az állomások, mert az átláthatatlan káoszt eredményezne. Az erre szolgáló **átvitel-vezérlésnek** lényegében két formája alakult ki.

- **CSMA/CD**-nek (Carrier Sense Multiple Access / Collision Detection) nevezzük azt a rendszert, melyben minden állomás (multiple access) folyamatosan figyeli az átvivő közeget (carrier sense) és ellenőrzi annak „tisztaságát", azaz azt, hogy nem ad-e már éppen valaki. Egy adó csak akkor kezdeményezhet üzenettovábbítást, ha a csatornát éppen senki sem használja. Előfordulhat persze, hogy két, vagy több adó gondolkodik egyformán, de ilyenkor sincs nagy baj, hiszen minthogy figyelik a csatornát, azonnal észreveszik az ütközést (collision detection), és megismételhetik az adást. De ha egyformán működnek, akkor egyformán ismétlik, és akkor a gond nem csökken. A CSMA/CD technika úgy küszöböli ezt ki, hogy az ütközést érzékelt adók nem azonnal, hanem – a további ütközést megelőzendő – véletlenszerű idő múlva kezdik újra adásukat. Az ilyen rendszer tipikus képviselője a helyi hálózatok között napjainkban leginkább elterjedt Ethernet.
- **Token Passing**, azaz a vezérjel átadás alapján működik például az Arcnet, Token Ring és az FDDI hálózat. Az állomások ebben az esetben – fizikai elhelyezkedésüktől függetlenül – egy logikai gyűrű mentén helyezkednek el. Az adók egy speciális üzenetet, a token-t (vezérjelet) adják körbe körbe, és az üzenet csak ehhez fűzhető. Mivel a token egyszerre csak egy állomás birtokában lehet, automatikusan kizárható az ütközés lehetősége.

4.2.2. Közös nyelv (protokoll)

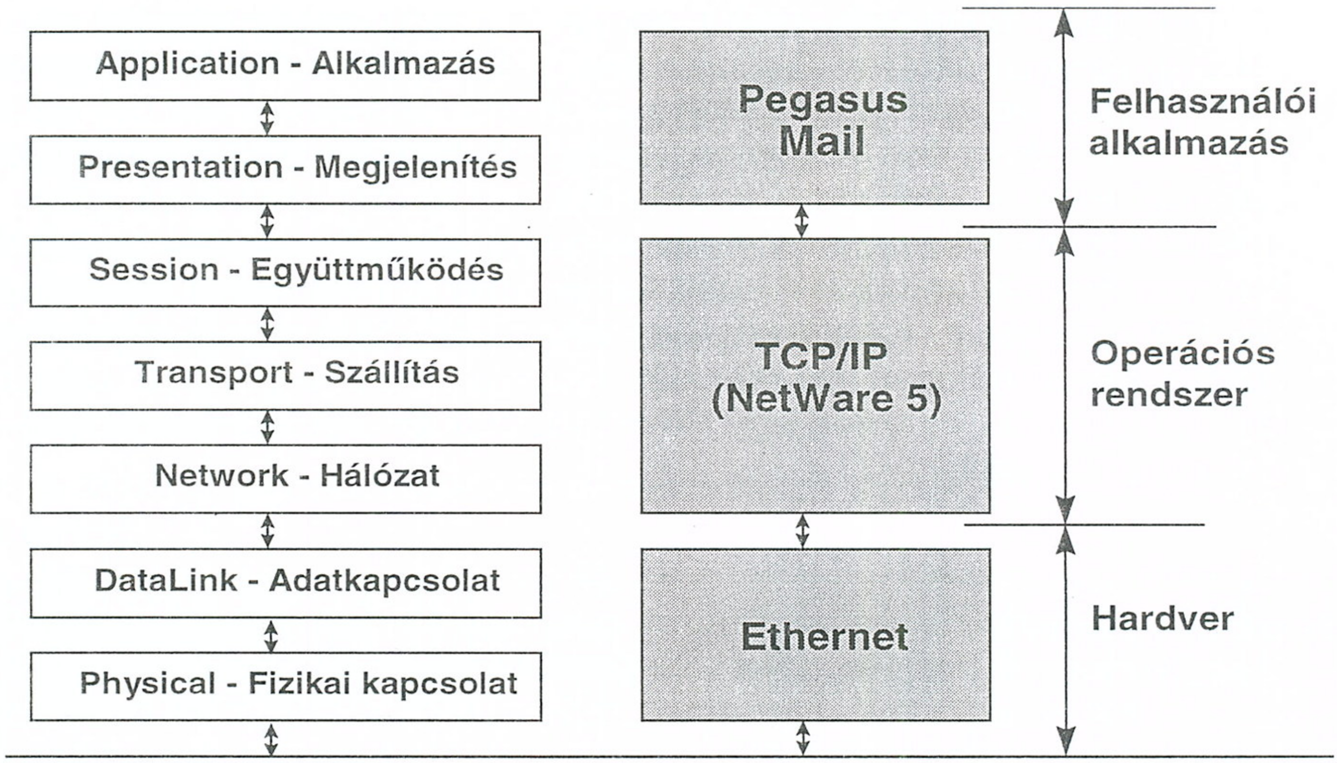
A szerencsére olyan intenzíven kezdtek terjedni, hogy a hardver és szoftver gyártók elemi érdeküknek érezték, hogy olyan egységes rendszereken munkálkodjanak, amelyek más berendezésekkel is jó együtt tudnak működni. Az erőfeszítés eredményeként létrejött egy nemzetközi ajánlás, amit **ISO-OSI** (International Standard Organisation – Open System Interconnection) néven ismerünk, amely vezérfonalul szolgált minden további fejlesztéshez. A szabvány egy réteges felépítésű rendszert definiál, minden réteghez meghatározott feladatot rendel.

Az egyes rétegek szabványos felületen kommunikálnak a szomszédos rétegekkel, valamint a másik gép ugyanazon rétegével. Az azonos rétegek kapcsolati szabályainak összességét nevezzük **protokoll**nak, illetve ezek összefüggő rendszerét **protokoll család**nak (gyakran előforduló protokoll családok például az IPX/SPX, a TCP/IP, az SNA, és a DECNET).

A gyakorlatban elterjedt rendszerek persze nem követik „szó szerint" a szabvány ajánlásait, egy-egy protokoll több réteget is megvalósíthat, egy rétegben több protokoll is helyet kaphat, illetve az is előfordul, hogy egy réteg csak félig-meddig valósul meg egy protokoll által.

Nézzünk egy példát!

A következő ábrán három számítógép réteges hálózati felépítése látható. A bal oldalon az eredeti, és a magyar elnevezésekkel, sötétebb árnyalattal jelölve pedig egy Novell hálózaton levelezésre használt számítógép példáján. A képen az is látható, hogy egy hálózati alkalmazás a hardver, operációs rendszer és felhasználói szoftver szintet egyaránt átöleli.



V.11. sz. ábra: Az ISO-OSI ajánlás, ill. egy megvalósulása

Vizsgáljuk meg egy levél elküldésének példáján a modell működését!

1. Megírom a levelet a levelező program beépített szövegszerkesztője segítségével (Felhasználói alkalmazás - Alkalmazás réteg - Pegasus Mail)
2. Ellátom digitális aláírással, titkosítom, hogy az illetéktelenek ne nézegethessék (Felhasználói alkalmazás - Megjelenítés réteg - Pegasus Mail)
3. Kapcsolatba lépek azzal a számítógéppel, amelyik majd továbbítja a levelemet (Operációs rendszer - Együttműködési réteg - Novell TCP/IP)
4. Megbízható, nyugtázott átvitelt kérek (Operációs rendszer - Szállítási réteg - Novell TCP/IP)
5. Megkeresem és megadom a célállomás címét, valamint a hozzá vezető útvonalat (Operációs rendszer - Hálózati réteg - Novell TCP/IP)
6. Megfelelő méretű darabokra tördelem a levelet, és előkészítem a legközelebbi megálló (azaz a végső útvonal felé vezető útvonal első állomása) felé történő továbbításra (Hardver - Adatkapcsolati réteg - Ethernet kártya)

7. Feszültség-impulzusokká alakítom a levelemből készült csomagok bitjeit és továbbítom őket a fizikai csatornán (Hardver – Fizikai réteg – Ethernet kártya)

Az ellenállomáson minden ugyanígy lépésről lépesre, de természetesen minden visszafelé, játszódik le.

Éppúgy, mint az emberi nyelvek, vagy a programozási nyelvek esetén a számítógépek is beszélhetnek több nyelvet, azaz ismerhetnek több protokollt. Az alábbi példa egy olyan számítógép operációs rendszerének hálózati szerkezetét mutatja, mely egyszerre képes igénybe venni az Internet, egy Novell szerver és a Windows-os hálózat szolgáltatásait.

<table>
<tr><td>Windows nyomtató megosztás</td><td>Novell fájlrendszer</td><td>Netscape WEB böngésző</td><td>Unix terminál emuláció</td></tr>
<tr><td>NetBIOS</td><td>IPX/SPX</td><td colspan="2">TCP/IP</td></tr>
<tr><td colspan="4">Ethernet</td></tr>
</table>

V.12. sz. ábra: Munkaállomás hálózati rétegei

4.2.3. Címek

Ahhoz, hogy a kapcsolat teljessé váljon, szükség van egyedi címekre is. Mivel minden réteg egy másik ugyanolyan réteggel kommunikál, ilyen cím több is lehet. A leggyakoribb címtípusokat az előző fejezet konkrét példáján mutatjuk be a fizikai rétegtől az alkalmazási réteg felé haladva.

4.2.3.1. Fizikai címek

A helyi hálózatokon legjobban az ethernet hálózatok használatosak. Ebben az esetben a fizikai cím egy gyárilag beállított, a felhasználó által nem változtatható hat bájtos szám, amit – mint általában mindig – hexadecimális alakban szoktunk megadni. Például:

`0000C0 - AB7416`

A hexadecimális számjegyek csoportosítása nem kizárólag a könnyebb olvashatóságot szolgálja. Az első hat számjegy a gyártó kódja (ebben a példában az SMC), egyediségére nemzetközi szervezetek ügyelnek, a második hat számjegy egyedi azonosító.

Más, kevésbé használt hálózattípusoknál előfordul, hogy a kártyán lévő kapcsolóval állítható a fizikai cím, de ennek alkalmazása nagyobb hálózatoknál feltétlenül kerülendő. A fizikai cím másik szokásos elnevezése a MAC (Media Acces Control) cím.

4.2.3.2. IP címek

A TCP/IP protokoll családban használatos címek 4 bájtos bináris számok.

Ez óriási nagy szám, hiszen 32 biten több mint 4 milliárd számítógép címezhető meg, de egyrészt a címtartomány nem teljesen kihasználható, másrészt ha elképzeljük, hogy minden háztartási gép belátható időn belül címet kaphat máris nem tűnik olyan soknak. A közeljövőben elterjedő, IPv6 szabvány 128 bites címmezője hoz várhatóan hosszú távú megoldást

A címet két részre oszthatjuk. Az első rész magának a hálózatnak a címe, a második pedig az egyedi gépé. Ha a gép címe helyén csupa egyest találunk (broadcast), azt jelenti, hogy az üzenet a hálózat minden állomására vonatkozik. A csak nullából álló cím tesztelésre szolgál, az üzenet az ilyen címet használó gépről nem jut ki.

Háromféle címosztály létezik:

A	0	Háló ($2^7 = 128$)			Gép ($2^{24} = 16$ millió)
B	1	0	Háló ($2^{14} = 16$ ezer)		Gép ($2^{16} = 64$ezer)
C	1	1	0	Háló ($2^{21} = 2$ millió)	Gép ($2^8 = 256$)

V.13. sz. ábra: „Klasszikus" IP címosztályok

Természetesen ezeket a címeket sem lenne kellemes kettes számrendszerben ábrázolni. Az IP címek körében a bájtokat egy-egy ponttal elválasztott tízes számrendszerbeli számmal adjuk meg. Például:

```
1100001 11100000 10001101 11110000 -> 193.224.141.240
```

Ez az ábrázolás már barátságosabb, de még mindig elég nehezen megjegyezhető. A probléma megoldása érdekében alakították ki az úgynevezett tartománynév rendszert (DNS - Domain Name System), mely legegyszerűbb megközelítésben olyan számítógépek (Name Server) rendszere, melyek egy név alapján táblázataikban megkeresik a névhez tartozó számot, és azt visszajuttatják a kérőnek. Az előző példában (vegyük észre, hogy az első három bináris számjegy tanúsága szerint ez egy C osztályú cím) a 193.224.141 azonosítja a hálózatot, a 240 a számítógépet. Rendeljük hozzá a `gdf-ri.hu` a karaktersorozatot (domain name) a 193.224.141 címhez (és természetesen ezt jegyeztessük be a megfelelő szervezettel), és nevezzük el `erebus`-nak a gépünket (hostname), és ezt jegyezzük be saját rendszerünkben. Ettől kezdve, ha a világon bárhol az `erebus.gdf-ri.hu`-ra hivatkoznak, a legközelebbi Name szerver kitalálja, hogy ez a mi hálózatunk, a saját Name szerverünk pedig az üzenetet a megfelelő gépre irányítja (`erebus`). Tehát:

`193.224.141.240 → erebus.gdf-ri.hu`

4.2.3.3. Levelezési címek

Ha a számítógépet meg tudjuk címezni, már nem nehéz azt a felhasználót sem elérni, akinek gépen található a postafiókja. A hagyományos névkonvenció a következő (például):

`info@erebus.gdf-ri.hu`, illetve **`info@193.224.141.240`**

4.3. Hálózattípusok

Ma az egyes intézmények hálózatatinak többsége kapcsolatban áll egymással, egységes címrendszert és protokollt használnak, mégis érdemes megkülönböztetni a viszonylag kis kiterjedésű **lokális hálózatokat** (LAN – Local Area Network) a **nagy kiterjedésű hálózatoktól** (WAN – Wide Area Network), melynek szinte egyeduralkodó képviselője az Internet. (Régebben használatos volt a városi hálózat (MAN – Metropolitan Area Network) kategória is, ami a LAN és a WAN között helyezkedik el, azonban ez az egységesedés miatt egy időre háttérbe szorul és csak napjainkban kezd visszalopakodni a szaknyelvbe, igaz, jócskán megváltozott értelemmel.)

A LAN/WAN megkülönböztetés oka elsősorban a sebesség. A világhálózat által biztosított óriási erőforrás tömeg egy kis részét (azt, amit a leggyakrabban szeretnénk használni) célszerű fizikailag is közelebb hozni, hiszen könnyebb rövid távon nagy sebességű átviteli csatornát létesíteni, változtatni a hardver , illetve szoftver konfiguráción, a rendszergazdához fordulni segítségért, vagy a szomszéd szobába átmenni a kinyomtatott levélért. A LAN fizikai összeköttetések jóval nagyobb átviteli sebességet tesznek lehetővé (10 Mbps – 1000Mbps) speciális kábeleiken, mint az egyéb célra készült kommunikációs hálózatokat (telefon, kábel TV) is használó WAN kapcsolatok (10kbps – 2 Mbps).

Felvetődhet a kérdés, ha a sebesség ennyire fontos, a kapcsolat másik két összetevője, a címzés és az alkalmazott protokoll miért egyezik meg a két hálózattípus esetén? Az ok a kényelem szeretete! Az elsősorban Internet céljaira kidolgozott technológiák ugyan lassabbak, mint a lassan kiszoruló LAN technikák, viszont a menedzselés szempontjából óriási előnyt jelent az egységesítés. Az egységes technológia az elnevezésekben is megmutatkozik, az **Intranet** tulajdonképpen nem más, mint LAN fizikai összeköttetést megvalósító, annak jó menedzselhetőségét öröklő, de az Internettől ellesett protokollokat használó hálózat.

4.4. Munkamegosztás a hálózaton

Az egymással kapcsolatban lévő, kommunikáló számítógépektől tehát azt várjuk, hogy tegyék lehetővé **erőforrásaik megosztását**, **növeljék a rendszer megbízhatóságát**, illetve **biztosítsák a gyors kommunikáció lehetőségét** felhasználóik között. E feladatok megvalósítása során kialakult egy specializáció a számítógépek között, létrejöttek olyan gépek, amelyek felépítése, illetve a rajtuk futó szoftver egy bizonyos részfeladat ellátására tette őket különösen alkalmassá. Ezek a gépek részben vagy teljes egészében arra szakosodtak, hogy kiszolgálják a hálózat többi résztvevőjének igényeit. Ma a hálózatba kapcsolt gépek többsége a **szerver** (kiszolgáló) – **kliens** (ügyfél) modell szerint működik.

Gyakori, hogy a szervert azonosítják a nagy hardver erőforrásokkal rendelkező gépekkel, pedig nem ez a lényeg, hanem a feladat. Előfordulhat, hogy szappanos doboz méretű doboz teszi lehetővé egy jelentős méretű hálózatnak, hogy közösen használhassanak egy nyomtatót.

Melyek a legfontosabb szolgáltatások? (Az alábbi vázlatos felsorolásban a szolgáltatás nevét kerek zárójelek között követi az a hálózattípus, melyre jellemző, a rövid leírást pedig szögletes zárójelben a szolgáltatáshoz kapcsolódó jellegzetes protokollok sora zárja.)

- **Elektronikus levelezés** (LAN/WAN) Elektronikus üzenet továbbítása, esetleg csatolt állományokkal együtt [SMTP, POP3, TCP/IP]
- **Fájlszerver** (LAN) Célja állományok gyors elérése abból a célból, hogy azt a kliens memóriájába betölthessük (tehát *nem* elsősorban a másolás), program esetén futtathassuk. [IPX/SPX, TCP/IP]
- **Nyomtatószerver** (LAN) Nyomtatók közös használatát teszi lehetővé. A nyomtatók megosztásának célja általában a takarékosság, de nem mindig! Ha egy gépről háromnál több nyomtatótípust szeretnénk használni, nem is igen van más lehetőség. [IPX/SPX, TCP/IP, NetBIOS]
- **FTP szerver** (WAN) Fájlok másolása (elsősorban) távoli szerverek és kliensek között. [FTP, TCP/IP]
- **WEB szerver** (LAN/WAN) Hipertext (azaz további szövegekre, multimédia elemekre utaló hivatkozást tartalmazó) szövegek továbbítása. Napjainkban ez a legnépszerűbb Internet alkalmazás. [HTTP, TCP/IP]
- **Terminál szerver** (LAN/WAN) Egy számítógép minden erőforrása igénybe vehető így, beleértve a processzort és a memóriát is. A programok a terminál szolgáltatást biztosító számítógépen futnak, a kliens felé mindössze a képernyőre szánt adatok jutnak el, illetve a kliens által megjelenített látszólagos terminál billentyűzete vezérli a távoli gépet. [Telnet, RLOGIN, SSH, TCP/IP]
- **Adatbázis szerver** (LAN/WAN) Nagy adatbázisokban történő keresések leghatékonyabban úgy valósíthatók meg, ha a hálózaton csak a kérdés (pl. SQL parancs), illetve a válasz (kigyűjtött adatok) kerülnek átvitelre, és nem az egész adatbázis, illetve kezelő programja. Ezt a feladatot látja el az adatbázis szerver, manapság igen gyakran Web felületen keresztül.
- **Proxy szerver** (WAN) Tulajdonképpen a lassan elérhető világháló, és a helyi hálózatok közötti gyorsítótár. Segítségével a gyakran letöltött fájlok, web oldalak a lokális hálózat sebességén érhetők el. [HTTP, FTP, TCP/IP]

- **Tűzfal szerver** (WAN) A lokális hálózat és az Internet elválasztására létesített szolgáltatás (a proxy-hoz hasonlóan), azonban célja a biztonság. Szabályozni lehet a belülről kifelé szóló kéréseket, illetve a kívülről befelé folyó forgalmat. [HTTP, FTP, TCP/IP]

A legfontosabb hálózati funkciók használatára egy későbbi fejezetben találunk példákat.

Ellenőrző kérdések

?

1. Milyen program típusokkal találkozhatunk leggyakrabban egy személyi számítógépen? 130
2. Mi az operációs rendszer alapfeladata? 132
3. Mit különbség és az összefüggés a shell és a kernel között?
4. Mi a feladata a kernelnek a hardver kezelés tekintetében? 133
5. Hogyan, milyen módszerekkel egységesíthető a harver-kernel interfész? 134
6. Mi a Plug and Play technika lényege?
7. Milyen kapcsolat van a szoftver eredetű megszakítások, és a rendszermag között? 135
8. Ismertesse a rendszermag fő funkcióit!
9. Mi az állomány (fájl)?
10. Milyen fájl elnevezési lehetőségeket ismer? 136
11. Sorolja fel az állományok fő jellemzőit! 138
12. Mi a katalógus (directory)?
13. Milyen katalógus rendszert használnak a mai operációs rendszerek? 139
14. Mi a kötet (volume)?
15. Milyen kötet elnevezéseket használnak a DOS eredetű operációs rendszerek?
16. Írjon néhány példát az abszolút fájl hivatkozásra? 140
17. Mi a relatív fájl elérés lényege?
18. Ismertesse az aktuális katalógus, aktuális kötet fogalmát!
19. Mi a keresési útvonal? Mire használható? 142
20. Mit nevezünk (a fájlrendszerekkel kapcsolatosan) clusternek?
21. Hogyan töltődik be az operációs rendszer a lemezről?
22. Mi a Master Boot Record, és mi köze a Partíciós táblához?
23. Mutassa be egy DOS kötet felépítését! 144

24. Milyen előnyei és hátrányai vannak a láncolt fájl (FAT) elhelyezés alkalmazásának? Hogyan mérsékelhetők a hátrányok?
25. Mi a parancsértelmező feladata?
26. Milyen elemekből épül fel egy DOS parancssor? Mi az egyes részek szerepe?
27. Ismertesse a DOS legfontosabb, fájlokra vonatkozó parancsait!
28. Mik az ablakozó rendszerek (pl. Windows) előnyei?
29. Miért alakítanak ki az operációs rendszerek mellett alrendszereket is?
30. Melyek a programkészítés alapvető lépései?
31. Mit nevezünk API-nak?
32. Mi a fordítóprogram (compiler) feladata?
33. Milyen feladatot lát el a kapcsolat szerkesztő (linker)?
34. Mi jellemzi az integrált fejlesztő rendszereket?
35. Mi a feladata a betöltőnek (loader)?
36. Mi az összefüggés egy rendszer redundanciája és hatékonysága között?
37. Hogyan befolyásolja a redundancia a megbízhatóságot?
38. Mi szükséges (minimálisan) ahhoz, hogy két számítógép egymással kommunikálni tudjon?
39. Mit nevezünk csomagkapcsolásnak?
40. Milyen – számítógép-hálózatokban alkalmazott – átvitel-vezérlési technikákat ismer?
41. Ismertesse a CSMA/CD lényegét!
42. Mit nevezünk számítógép-hálózati protokollnak?
43. Mire vonatkozik az OSI modell, és milyen részekből áll?
44. Rajzoljon le egy konkrét hálózati rétegszerkezet megvalósítást!
45. Ismertesse az IP címek képzésére vonatkozó szabályokat!
46. Miért van szükség tartomány nevekre? Mi a DNS (Domain Name System)?
47. Mi a fő különbség a LAN és a WAN között?
48. Melyek a szerver–kliens modell alkalmazásának előnyei?
49. Mi az Intranet?
50. Ismertesse a számítógép-hálózatok alapfeladatait!
51. Melyek a legfontosabb számítógép-hálózati szolgáltatások?
52. Mi az összefüggés és a különbség a Tűzfal és a Proxy szerver között?
53. Mi az összefüggés és a különbség a Fájl és az FTP szerver között?
54. Mikor használhatók ki igazán a nyomtató szerver alkalmazásának előnyei?

VI. fejezet

AZ UTASÍTÁSKÉSZLET ÉS FELHASZNÁLÁSA A PROGRAMOZÁSBAN

Ebben a fejezetben először a mikroszámítógépek utasításszerkezetét és utasításkészletét tekintjük át, melynek során az Assembly szintű programozásban leggyakrabban használt tipikus utasításokat konkrét példákkal is bemutatjuk. Ezt követően tárgyaljuk azokat a fontosabb módszereket és eljárásokat, melyek – az utasításkészletet, mint a lehetséges műveletek szótárát használva – segítenek egyes algoritmusoknak a „gép nyelvére történő lefordításában", azaz a megfelelő programok előállításában.

Erre a célra egy elképzelt mikroszámítógépet fogunk megtervezni. Ez egyetlen létező mikroszámítógéppel sem lesz azonos, de segítségével képesek leszünk a valódi processzorok utasításkészletének legfontosabb sajátosságait modellezni. A **leegyszerűsített számítógépmodell** emellett az egyes utasítások végrehajtásának bemutatását is megkönnyíti és érthetőbbé teszi. Megkonstruálása során lehetővé teszi, hogy bepillantást nyerjünk a processzortervezés alapelveinek körébe.

A processzormodellünk megalkotásában az is fontos szempont volt, hogy a legelterjedtebben használatos Intel processzoroktól csak a legszükségesebb mértékben térjünk el. Ezért elképzelt processzorunk jó közelítéssel egy erősen egyszerűsített, primitív Intel processzorral működő számítógépnek tekinthető. (Az Assembly utasítások szimbolikus nevei is azonosak az IBM PC Assemblyben megfelelő mnemonikjaival.)

Jószerivel azt mondhatjuk, hogy utasításkészlet annyiféle van, ahány fajta processzor, sőt még azonos processzorcsaládon belül is lehetnek eltérések az utasításkészletek között (így például a legújabb Pentium III-as processzor utasításkészlete 70 utasítással bővebb elődjének, a Pentium II-es processzornak az utasításkészleténél.)

Egy konkrét processzor utasításkészlete esetenként több száz különböző fajta utasítást tartalmaz, ezek dokumentációja általában több ezer oldalas kézikönyvekben található meg.

Emiatt is célszerű az utasításkészlet tárgyalása során egy konkrét processzor helyett egy modellt vizsgálnunk. Ezzel szemben a következő követelményeket támasztottuk:

- az utasításkészlet tartalmazza az Assembly szintű programozásban leggyakrabban használt utasításokat;
- az összes olyan utasítást képesek legyünk a modell alapján bemutatni, melyeket a legtöbb 3-generációs programnyelv tartalmaz funkcionálisan (például összeadás, szubrutinhívás stb.);
- az utasításkészlet terjedelme olyan legyen, hogy az ezekből kiválasztott utasításokkal képesek legyünk gyakorlati programozási feladatokat megoldani;
- az Intel processzorcsaládtól és az IBM PC Assemblytől csak az egyszerűsítés, és az utasításvégrehajtás érthetőbb bemutatása érdekében térjen el a modell.

1. A mikroszámítógép modell felépítése

Az elképzelt mikroszámítógépünk természetesen azokból a részegységekből épül fel, melyeket a II. és III-as fejezetben már megismertünk:

- a processzor (CU, ALU, belső sín, regiszterek);
- a sínrendszer (adat, cím, vezérlőbusz);
- a főtár (memória, RAM);
- az I/O portok.

Egy konkrét processzor utasításszerkezetének meghatározásához és az utasításkészletének felépítéséhez azonban ez az általános architektúra önmagában még nem ad elegendő információt. Néhány fontosabb, eldöntendő kérdés:

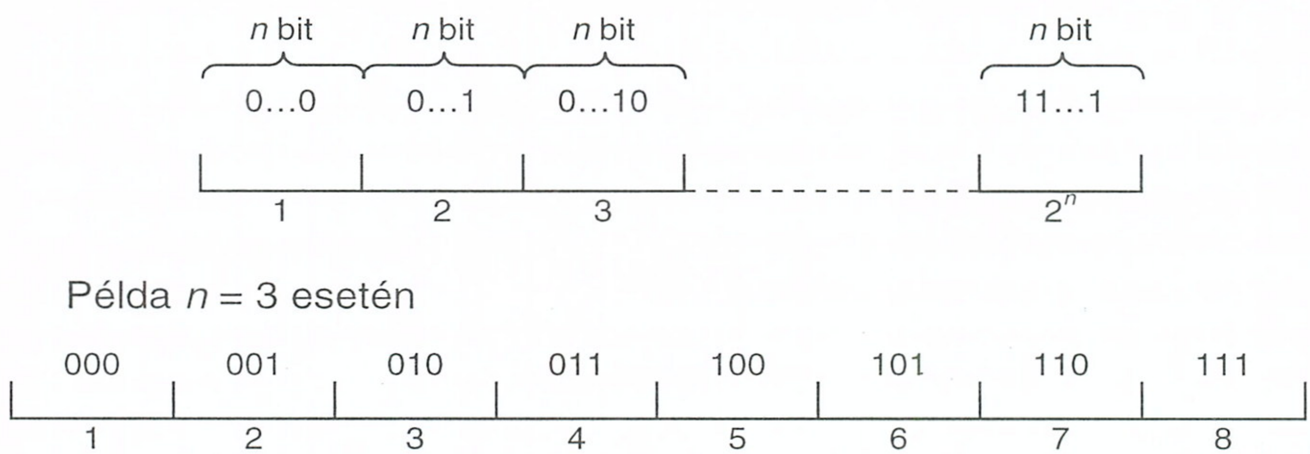
- Milyen feladatokra akarjuk a regisztereket felhasználni, összesen hány regisztert tartalmazzon a CPU, egy regiszter mérete mekkora (azaz hány bitből álljon) legyen?
- Összesen hány bájtot akarunk megcímezni a számítógép memóriájában, vagy másképp fogalmazva: mekkora legyen a főtár maximális kapacitása bájtban?
- A CPU milyen állapotjellemzőit (flag-ek) akarjuk rögzíteni az állapotregiszterben?
- Egy lépésben hány bitből álló címeket kell átvinni a címsínen?
- Egy lépésben hány bitből álló adategységeket kell mozgatni az adatsínen a processzor–memória, illetve a processzor és az I/O portok között?
- Milyen puffer és állapotregisztereket tartalmaznak az I/O portok?

Mint látni fogjuk, ezekre a kérdésekre adható válaszok nem mindig függetlenek egymástól. Erre egy fontos példa:

Ha már rögzítettük a főtár maximális kapacitását bájtban, akkor ezzel egyúttal azt is eldöntöttük, hogy

- hány bitesek a címeket tároló regiszterek,
- egy lépésben hány bites címeket kell átvinni a címsínen.

Tegyük fel ugyanis, hogy számítógépünk lehetséges maximális tároló kiépítése esetén a főtár összesen 2^n darab bájtot tartalmaz. Ekkor ezeket a bájtokat n bitet tartalmazó bináris számokkal tudjuk címezni, 0-tól 2^n-1-ig (lásd VI.1. sz. ábra).



VI.1. sz. ábra
2^n darab bájtot tartalmazó memória címzése

! Ha viszont **egy bájt megcímzéséhez *n*-bit szükséges, akkor egy lépésben ennyi bitet kell átvinni a címsínen és a címeket tároló regisztereknek is képesnek kell lenniük *n*-bites számokat befogadni**.

A napjainkban használatos Intel processzor alapú számítógépek címsínje egy lépésben 32 bites címek átvitelére képes (32 címvonalat tartalmaz). Ez egyúttal előző gondolatmenetünk alapján azt is jelenti, hogy ezekben a számítógépekben a maximális memóriakapacitás 2^{32} bájt = 4 Gigabájt lehet. Természetesen ez egy elméleti maximum, a gyakorlatban használt számítógépek memóriakapacitása általában a 8 és 128 Megabájt közötti érték.

Az elkövetkezendőkben modellünk konkretizálásával megadjuk a választ az előbbiekben feltett összes kérdésre.

1.1. Regiszterkészlet

Egy **processzor regiszterkészletének megtervezése** során egymással ellentétes irányba ható tényezőket kell figyelembe venni:

- Mivel a regiszterek a leggyorsabb tárolók, ezért célszerű, ha a processzor minél több regisztert tartalmaz, mert nyilvánvalóan ekkor lesz a leggyorsabb.
- Minden egyes regiszter a chip áramköri egységeiből meghatározott mennyiségűt lefoglal. Emiatt a regiszterek a legdrágább tárolóeszközök, így a sok regiszter processzor árát is jelentősen megnöveli.

Erre figyelemmel a processzorok tervezése során gondosan mérlegelni kell, hogy milyen feladatokra hány darab regisztert építünk be a CPU-ba.

Az elkövetkezendőkben modellprocesszorunk regiszterkészletéből az **általános regisztereket** fogjuk definiálni, azaz azokat a regisztereket, melyeket a gépi utasítások közvetlenül felhasználhatnak.

Elméletileg egy processzor már akkor is működőképes, ha csak három regisztert, egy akkumulátort, egy utasításszámlálót és egy állapotregisztert tartalmaz.

Ebben az esetben viszont például a verem címének kezelése, vagy a tömbök feldolgozásához szükséges indexelés csak a memóriában lenne megoldható. Ez nagyon bonyolulttá tenné a programokat és a processzort is lelassítaná. (A regiszterek hozzáférési ideje kb. tizedrésze a memóriának.)

Erre is figyelemmel a modell **számítógépünk regiszterkészletét** a következőkben határozzuk meg:

AX *Akkumulátor* (Accumulator)
BX *Bázisregiszter* (Basis register)
CX *Számláló regiszter* (Counter register)
DX *Adatregiszter* (Data register)
SP *Veremmutató* (Stack pointer)
BP *Bázismutató* (Basis pointer)
SI *S-Indexregiszter* (Source index)
DI *D-Indexregiszter* (Destination index)
IP *Utasításszámláló* (Instruction pointer)

Itt az AX, BX, CX, DX, SP, BP, SI, DI, IP a regiszterek szimbolikus rövid nevét jelöli. Az egyes regiszterek felhasználását a programokban a későbbiek során fogjuk bemutatni.

Az egyszerűség és a jobb szemléltethetőség érdekében a modellünkben szereplő **regiszterek hosszát egységesen 16 bit-ben határozzuk meg**.

Ebből néhány dolog azonnal következik:

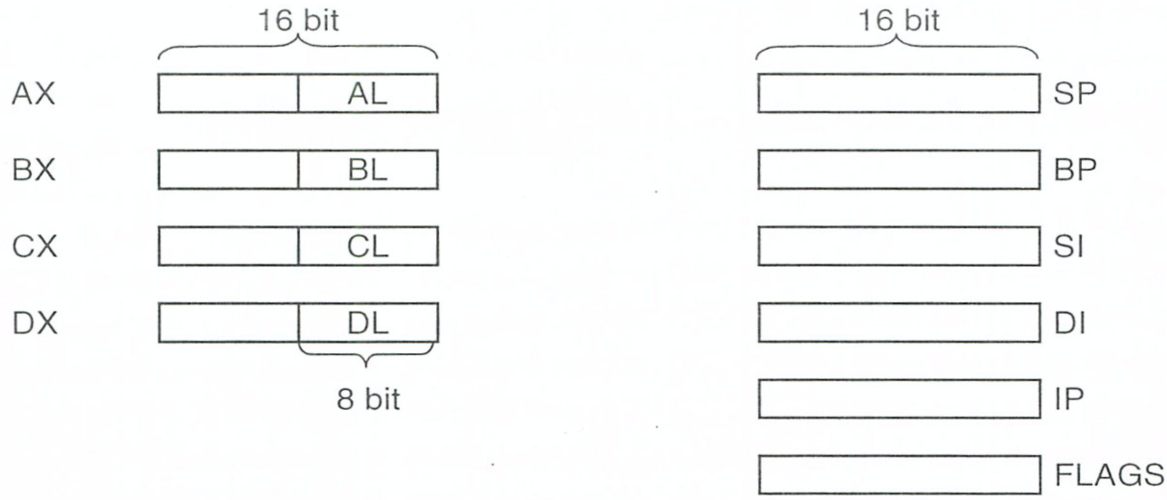
- A processzor maximum 16-bites, illetve 15-bites számokkal képes előjel nélküli, illetve előjeles műveleteket elvégezni;
- Számítógépünk maximális memóriakapacitása 2^{16} bájt = 64 Kilobájt;
- Számítógépünk címsínje 16 címvonalat tartalmaz.

Ha 16-bites regisztereink vannak, akkor célszerű biztosítani a memóriából **16-bites (2 bájtos) szavak kiolvasását illetve visszaírását**.

Feltételezzük, hogy a főtár memóriavezérlő áramkörei a szavas hozzáférés mellett az **1 bájtos memóriaműveleteket** is lehetővé teszik. Ebben az esetben a sínrendszer úgy viszi át, a processzor pedig úgy fogadja a 8-bites bájtokat, hogy azok a regiszterek alsó legkisebb helyi értékű 8 bit-jén kerülnek letárolásra.

Ezeket a bájtokat az AX, BX, CX, DX regiszterekben AL, BL, CL, DL-el fogjuk jelölni, és az 1 bájtos memóriaműveleteket csak e regiszterekre engedjük meg.

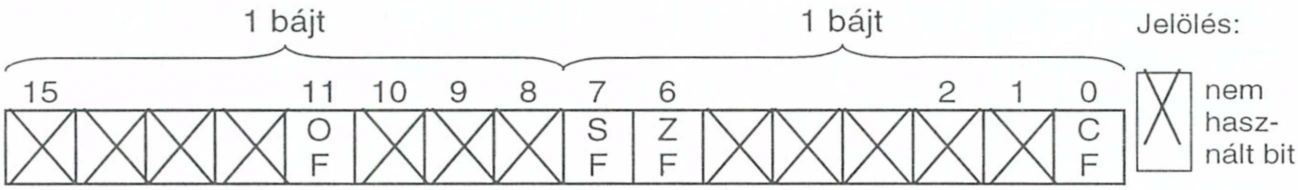
A bevezetett regiszterkészletet és az 1.2. pontban tárgyalt FLAGS állapotjelző regisztert foglalja össze a VI.2. sz. ábra.



VI.2. sz. ábra
A modellprocesszor regiszterkészlete

1.2. Állapotjelzők

A **FLAGS állapotjelző regiszter bitjei** különösen a feltételes ugróutasítások kivitelezése szempontjából fontosak. Ez a regiszter modellünkben szintén 16 bitet fog tartalmazni a VI.3. sz. ábra szerint.



VI.3. sz. ábra
A FLAGS regiszter felépítése

A VI.3. sz. ábra jelöléseivel az egyes flag-ek jelentése a következő:

- **CF (Carry flag)** Értéke 1, ha egy művelet eredményének legmagasabb helyi értékű (2^{15}) bitjén átvitel keletkezik
- **ZF (Zero flag)** Értéke 1, ha egy művelet eredményének minden bitje 0
- **SF (Sign flag)** Megegyezik a művelet eredményének legmagasabb helyi értékű bitjével (azaz a kettes komplemens ábrázolás szerinti értéke 1, ha az eredmény negatív)

- **OF (Overflow flag)** Előjeles értékkel végzett műveleteknél a túlcsordulást jelzi, azaz OF = 1, ha az eredmény nem előjelhelyes

Természetesen egy valódi processzorban a FLAGS regiszter általunk „nem használt"-nak minősített bitjei további fontos információkat tartalmaznak a processzor állapotáról, illetve vezérlik a processzor működését. Ilyen például: a paritáshiba jelzése, a hardver megszakítások letiltása stb.

1.3. I/O portok

Mikroszámítógép modellünk nagyon hiányos lenne, ha nem biztosítaná az **input/output műveletek** végrehajthatóságát.

Ahhoz, hogy az utasításkészletben input/output utasításokat is szerepeltetni tudjunk, konkrétan specifikálni kell azokat az I/O portokat, melyekhez a perifériák kapcsolódhatnak.

Modellünkben két I/O portot definiálunk.

- *0-s port*, mely input adatok beolvasására szolgál,
- *1-es port*, mely output adatok továbbítására szolgál.

Az egyszerűség kedvéért képzeljük azt, hogy a 0-s port a billentyűzetről érkező adatokat fogadja, az 1-es port pedig az adatokat a monitor képernyőjére történő kiíráshoz továbbítja.

Minden port modellünkben két 8-bites regisztert tartalmaz. Ezek:

- a *pufferregiszter*, mely az adatok átmeneti tárolására szolgál;
- az *állapotregiszter*, mely a port állapotinformációit tárolja.

A következőkben megadjuk az állapotregiszterek egyes bitjeinek értelmezését:

- **A 0-s port állapotregiszterében**

 2^0 bit = 1, ha a port pufferregiszterébe új adat érkezett az input eszközről

 = 0, a port pufferregisztere változatlan, azaz nem érkezett új adat

 2^1 bit = 1, ha a port pufferregiszterét a CPU még nem olvasta ki

 = 0, ha a port pufferregiszterét a CPU már kiolvasta

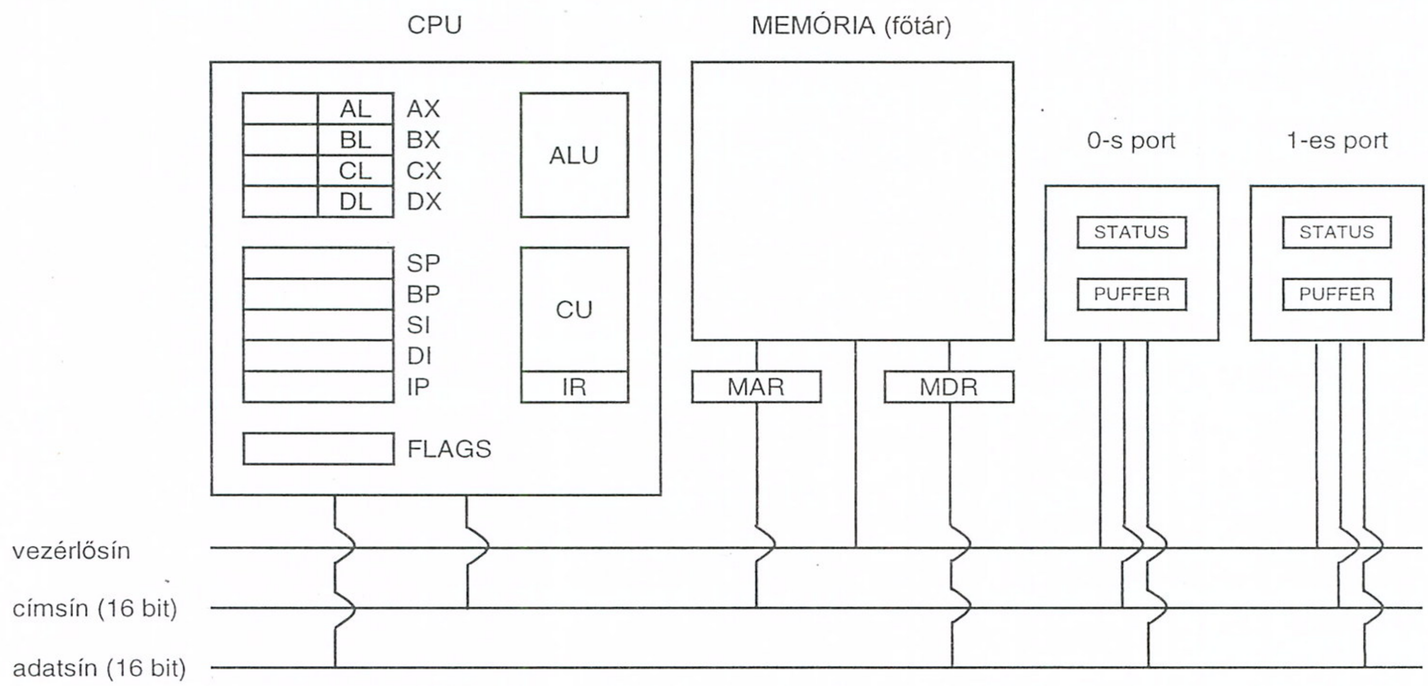
- **Az 1-es port állapotregiszterében**

2^0 bit = 1, ha nincs az output eszközhöz továbbítandó adat az 1-es port pufferében
= 0, ha az output eszközhöz továbbítandó új adat van az 1-es port pufferében

2^1 bit = 1, ha az output eszközhöz való adattovábbítás rendben befejeződött, a port felkészült új adat fogadására
= 0, ha az output eszközhöz való adattovábbítás még nem fejeződött be

Az állapotregiszterek 0-s értékű bitjeit a CPU, az 1-es értékű bitjeit pedig a külső eszközök állítják be. Nyilvánvaló, hogy az input/output művelet végrehajthatóságának a CPU oldaláról az a feltétele, hogy az állapotregiszter 2^0 és 2^1 helyi értékű bitjei 1-es értékűek legyenek.

Az előző specifikációk alapján most már felvázolhatjuk **modellszámítógépünk** pontos **architektúráját** (lásd VI.4. sz. ábra).



VI.4. sz. ábra
A modellszámítógép felépítése

A korábbiakban tárgyalt regisztereken kívül az ábrán még három **rendszerregisztert** is feltüntettünk. Ezek

- az *IR utasításregiszter* (Instruction Register), mely a memóriából kiolvasott programutasítás átmeneti tárolására szolgál a CPU-ban;

- a *MAR memória címregiszter* (Memory Adress Register), mely a memóriából kiolvasandó adatok címét tárolja;
- a *MDR memória adatregiszter* (Memory Data Register), mely a memóriába beírandó, illetve onnan kiolvasandó adatok átmeneti tárolója.

Mivel 1, illetve 2 bájtos adategységeket akarunk mozgatni a CPU és a főtár között, nyilvánvaló, hogy MDR-nek 16-bites regiszternek kell lennie. Címeink 16 bitesek, ezért MAR is 16-bites regiszter. IR méretét nyilván az határozza meg, hogy mekkora egy gépi utasítás maximális hossza. Mint ezt a 3. pontban le fogjuk vezetni, ez 32 bit, ennek megfelelően IR modellünkben legyen 32-bites regiszter.

Az IR, MAR, MDR rendszerregiszterek, melyek egy gépi utasítás végrehajtásán belüli elemi lépésekben játszanak fontos szerepet. Ezek a programutasításokkal nem elérhetők, nem címezhetők és tartalmuk sem változtatható meg.

Mint már említettük, a memóriához azért kell pufferregisztereket alkalmazni, mert a CPU működése lényegesen gyorsabb (magasabb az ütemező órajel frekvenciája) mint a memóriáé.

2. A modellprocesszor utasításszerkezetének és készletének megtervezése

2.1. Címzési módok

A **modellprocesszornak címzési módja**inak kidolgozása során arra a kérdésre kell megkeresnünk a választ, hogy a gépi utasítás-végrehajtás különböző „szereplőit":

- az *általános regisztereket,*
- a *memória 1, illetve 2 bájtos egységeit,*
- az *I/O portok regisztereit,*

milyen módon tudjuk azonosítani a programutasításokban. Ezekhez a számítógép bináris működése miatt nyilvánvalóan bináris kódokat kell hozzárendelni és egy kódnak egyértelműen azonosítani kell egy tárolóegységet.

A **regisztereket** egyszerűen 4-bites **bináris sorszámukkal címezzük** a következő felsorolás szerint:

Binárisan	Decimálisan	Hexadecimálisan	Szimbolikus nevük
0000	0	0	AX
0001	1	1	AL
0010	2	2	BX
0011	3	3	BL
0100	4	4	CX
0101	5	5	CL
0110	6	6	DX
0111	7	7	DL
1000	8	8	SP
1001	9	9	BP
1010	10	A	SI
1011	11	B	DI
1100	12	C	IP

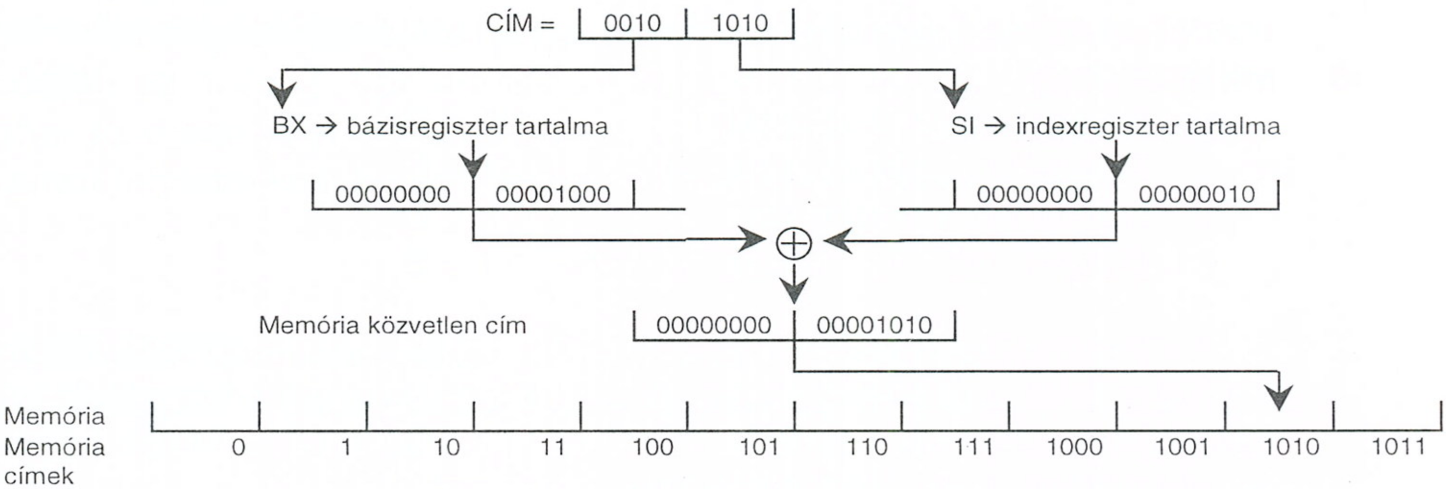
VI.5. sz. ábra
A regiszterek bináris kódjai

Felvetődhet a kérdés, hogy a FLAGS regisztert miért nem láttuk el egy bináris azonosítókóddal. Ennek az az oka, hogy ezt a regisztert közvetlenül a gépi utasításokban nem lehet megcímezni. Ugyanakkor külön utasításokkal a FLAGS regiszter tartalmához is hozzáférhetünk.

A **memória címzéséhez háromféle módot engedünk meg** processzor modellünkben. Ezek:

- a *regiszter indirekt címzése*, ekkor az utasításban szereplő regiszter tartalmazza azt a 16-bites memóriacímet, ahol az adat megtalálható;
- a *közvetlen adatcímzés*, amikor magában az utasításban található meg egy 8-, illetve 16-bites adat- vagy címérték;
- a *bázis-relatív és indexelt címzés*, amikor az utasítás két regiszter címét tartalmazza.

Ebben az esetben az adat tényleges 16-bites memóriacímét úgy számítjuk ki, hogy a két regiszterben található címet összeadjuk (lásd a VI.6. sz. ábra).



VI.6. sz. ábra
A bázis-relatív és -indexelt címzés

Végezetül az **I/O portok címzését** határozzuk meg, melynél figyelembe kell venni, hogy a portban található két regisztert önállóan meg kell címeznünk az utasításokban. Ezért e portok regisztereinek a címeit a következőképpen definiáljuk:

- 0 port puffer regiszter 00
- 0 port állapotregiszter 01
- 1 port puffer regiszter 10
- 1 port állapotregiszter 11

Foglaljuk össze, hogy a programutasításokban szereplő tárolóelemeket milyen kódokkal azonosítjuk:

a/ Általános regiszterek	4-bites regiszterkód
b/ Memória indirekt címzése regiszterrel	4-bites regiszterkód
c/ Közvetlen adatcímzés	8- és 16-bites adatérték
d/ Memória bázis-relatív és -indexelt címzése	két 4-bites regiszterkód
e/ I/0 portok regisztereinek címzése	2-bites port kód

Vegyük észre a következő problémákat:

- Ha az utasításban egy operandus helyén egy regiszter kódja van, hogyan ismeri fel a processzor, hogy a regiszter egy adatot vagy egy címet tartalmaz (Másként fogalmazva, mi alapján lehet elkülöníteni az a/ és b/ esetet?);
- Ha az utasítás egy 8 bites bináris számot tartalmaz, mi alapján dönthető el, hogy ez egy közvetlen adatcímzéshez tartozó bájt vagy pedig két regiszter azonosítója (c/ és d/ eset).

A többértelműség kiküszöbölése érdekében az a/, b/, c/, d/ eseteknek megfelelő **címzési módokat binárisan fogjuk kódolni** a következőképpen:

- Regiszterben lévő adat 00 = R
- Regiszter indirekt címzése 01 = I
- Közvetlen adatcímzés 10 = D
- Bázis-relatív és -indexelt címzés 11 = B

Az elkövetkezendőkben az a), b), c), d) címzési módokat rendre R, I, D, B-vel fogjuk jelölni. Az I/0 port regisztereinek a címzését külön egyedi esetként kezeljük, mivel ezt az I/0 utasítás kódja egyértelműen azonosítja (lásd a 4. pont).

2.2. A gépi utasítás szerkezete

Mint később látni fogjuk modellprocesszorunk egy gépi utasítása bináris formában és Assembly mnemonikkal a VI.7. sz. ábra szerinti formátumú.

Assembly formátum:	ADD AX, [BX+SI]
Gépi kód:	0000 0000 0010 1010

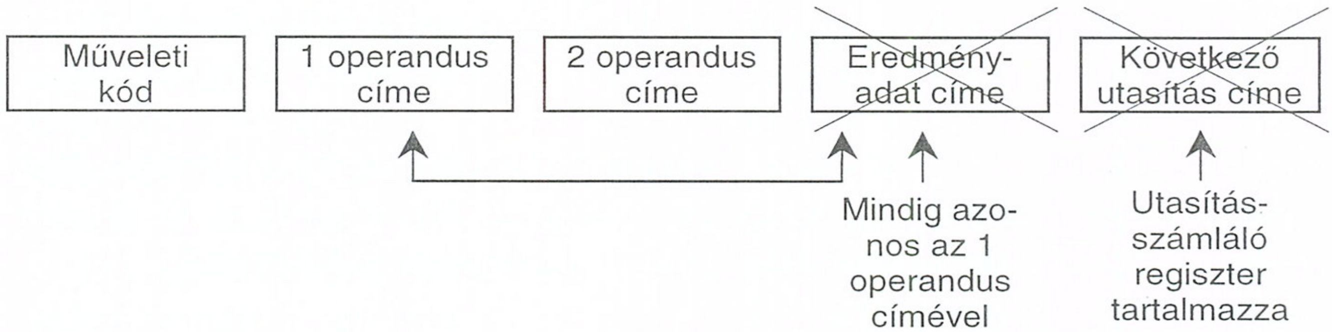
VI.7. sz. ábra
Bázisindexelt címzésű két bájtot az akkumulátorhoz hozzáadó utasítás

Figyeljük meg az ábrán látható bitsorozatot és tegyük fel a kérdést: honnan állapítja meg a processzor, hogy a bitek melyik csoportja jelenti a műveleti kódot (mit kell elvégezni) és az utasításban résztvevő tárolók címét?

Nyilvánvalóan, ehhez egy egyértelmű és konzekvensen alkalmazott szabályrendszer kell, mely meghatározza, hogy az utasításokban lévő biteket a processzor hogyan értelmezze. Ezt határozza meg a **gépi utasítás szerkezete**, melyet most fogunk kidolgozni modellprocesszorunk számára.

Egy gépi utasítás funkcionális részekre való felbontását az határozza meg, hogy a processzornak milyen információra van szüksége az utasítás végrehajtásához.

Ha egy gépi utasításban maximum két adat vehet részt, akkor a művelet végrehajtásához a processzornak ismernie kell a művelet kódját, az 1 és 2. operandus címét, annak a tárolóhelynek a címét ahol az eredményt el kell helyezni, valamint azt a címet, ahol a programot folytatni kell, ha az utasítás végrehajtása befejeződött.



VI.8. sz. ábra
A gépi utasítás logikai szerkezete

A következő utasítás címe és az eredményadat címe részeket kihagyhatjuk a gépi utasításból a következő két konvenció bevezetésével:

- *a következő utasítás címét mindig az IP utasításszámláló regiszter tartalmazza;*
- *két operandust tartalmazó gépi utasításoknál az eredményadatot mindig az első operandus címén* (annak korábbi állapotát felülírva) *kell elhelyezni.*

Ezt figyelembe véve modellszámítógépünk utasításkészletében maximum kétcímű utasításokat fogunk szerepeltetni.

Az utasításszerkezet specifikálása során az lesz a célunk, hogy a megtervezett utasítások minél rövidebbek legyenek, tehát ahol ez lehetséges takarékoskodni fogunk a bitekkel. Ennek az az értelme, hogy a rövidebb gépi utasításokból álló program egyrészt kevesebb tárolóhelyet igényel, másrészt végrehajtása is gyorsabb, mivel az utasítások kiolvasása során kisebb mennyiségű adatot kell mozgatni a főtár és a processzor között.

Az előbbieket figyelembe véve modellprocesszorunk gépi utasítása négy, három illetve két főrészből fog állni (lásd a VI.9. sz. ábra).

Kétcímű utasítás	Műveleti kód	Címzési mód	1. operandus címe	2. operandus címe
Egycímű utasítás	Műveleti kód	Címzési mód	Operandus címe	
0 című utasítás	Műveleti kód			

VI.9. sz. ábra
A modellprocesszor utasításainak fő részei

Próbálkozzunk most azzal, hogy a műveleti kód és a címzési mód mezőkre ne használjunk el többet 1 bájtnál. Mivel az R, I, D, B címzési módok 2 biten kódolhatóak, ezért a **címzési mód mező**

- kétcímű utasításnál 4 bit,
- egy című utasításnál 2 bit,
- 0 című utasításnál 0 bit

tárolóhelyet igényel. Ebben az esetben a műveleti kód kódolására

- kétcímű utasításnál 4 bit,
- egycímű utasításnál 6 bit,
- 0 című utasításnál 8 bit

lehetőségünk marad.

A műveleti kódot OP-vel, a címzési módot AM jelölve ezt mutatja be a VI.10. sz. ábra.

Bitpozíció	7	6	5	4	3	2	1	0
Kétcímű utasítások	OP				AM			
Egycímű utasítások	OP						AM	
0 című utasítások	OP							

VI.10. sz. ábra
A műveleti kód és a címzési mód mezők felépítése

Ennek megfelelően az utasításkészletben például maximum 16 (2^4) kétcímű utasítást leszünk képesek kódolni. Ha ezekbe a keretekbe az utasításkészletbe kiválasztott konkrét utasítások beleférnek, akkor a műveleti kód és a címzési mód bináris kódolását 1 bájton meg tudjuk oldani.

Tekintsük most át utasításaink címrészének felépítését. Ehhez emlékeztetünk arra, hogy a címzési mód szerint:

- R 4 bit
- I 4 bit
- D 16 bit (8 bit)
- B 4 + 4 = 8 bit

tárolóhelyre van szükségünk.

Itt beleütköztünk abba a problémába, hogy 8- és 16-bites adategységeket is szeretnénk mozgatni a főtár és a CPU között. Hogyan lehet ezt kifejezni az utasításszerkezetben, mivel a kódolásra felhasználható bitjeink már „elfogytak”.

Ezt az ellentmondást a következő konvencióval küszöbölhetjük ki:

Ha az 1-es operandusban 16- illetve 8-bites regiszter szerepel, akkor ennek megfelelően a 2-es operandus által meghatározott memóriacímről 2 illetve 1 bájtot kell átvinni a CPU-ba.

Az **utasítások címrészének felépítése szerint elvileg a következő utasítástípusok lehetségesek:**

- *Kétcímű utasítások:* RR, RI, RD, RB,
IR, II, ID, IB,
BR, BI, BD, BB
- *Egycímű utasítások:* R, I, D, B
- *0 című utasítások*

Ehhez két megjegyzést célszerű tenni:

- Az utasítástípusok most még csak „elméletileg” léteznek, ezeket a variációkat majd a konkrét utasítások definiálásakor töltjük fel tartalommal.
- A DR, DI stb. utasítástípusok nyilvánvalóan nem értelmesek, mert konvenciónk szerint az utasítás eredménye az 1-es operandusban keletkezik.

A modellprocesszorunk utasításszerkezetének teljeskörű definícióját a VI.11. sz. ábra tartalmazza.

Utasítás-típus	1 bájt		1 bájt		1 bájt		1 bájt		
RR	OP	00 00	R1	R2					KÉT CÍMŰ
RI	OP	00 01	R1	R2					
RD	OP	00 10	R1	0000 0000	DB DW				
RB	OP	00 11	R1	0000	R2	R3			
IR	OP	01 00	R1	R2					
II	OP	01 01	R1	R2					
ID	OP	01 10	R1	0000 0000	DB DW				
IB	OP	01 11	R1	0000	R2	R3			
BR	OP	11 00	R1	R2	0000	R3			
BI	OP	11 01	R1	R2	0000	R3			
BD	OP	11 10	R1	R2	DB DW				
BB	OP	11 11	R1	R2	R3	R4			
R	OP	00	0000	R1					EGY CÍMŰ
I	OP	01	0000	R1					
D	OP	10	DB DW						
B	OP	11	R1	R2					
0 című utasítás	OP								0 című

Jelölések:

(ferdén vonalkázott)	= Műveleti kód	R1, R2, R3, R4	= Az utasításokban szereplő regiszterek
(világos pontozott)	= 1-es operandus	DB	= 1 bájtnyi adat
(sötét pontozott)	= 2-es operandus	DW	= 2 bájtnyi adat

VI.11. sz. ábra
A modellprocesszor utasításszerkezetének definíciója

Az ábra alapján látható, hogy
- a kétcímű utasítások 2, 3 és 4 bájt,
- az egycíműek 2, 3 bájt
- a 0 címűek 1 bájt

hosszúak.

2.3. Az utasításkészlethez tartozó utasítások kiválasztása és kódolása

Az elkövetkezendőkben sorra vesszük a számítógép különböző utasításcsoportjait és a funkciókat értékelve meghatározzuk, hogy modellprocesszorunk utasításkészletében milyen utasításokat szeretnénk szerepeltetni. A kiválasztásnál arra törekszünk, hogy a leggyakrabban használt Assembly utasítások szerepeljenek az utasításkészletben és ezekből olyan választék legyen, mely programok megírását is lehetővé teszi.

Aritmetikai és logikai utasítások

Az **aritmetikai műveletek** közül csak a legszükségesebbeket választjuk ki. Ezek:
- ADD *összeadás* (fixpontos)
- SUB *kivonás* (fixpontos)
- CMP *összehasonlítás*
- INC *inkrementálás, azaz 1-gyel növelés*
- DEC *dekrementálás, azaz 1-gyel csökkentés*
- NEG *komplementálás, azaz (-1)-gyel való szorzás.* (kettes komplemens)

A **logikai műveletek** közül négyet szerepeltetünk modellprocesszorunk utasításkészletében. Ezek:
- AND *logikai „ÉS”*
- OR *logikai „VAGY”*
- XOR *logikai kizáró „VAGY”*
- NOT *logikai „NEM”* (egyes komplemens)

Adatmozgató utasítások

Az adatmozgató utasításokat úgy fogjuk kiválasztani, hogy segítségükkel

- a memória és regiszter
- a regiszter és regiszter
- a regiszter és verem
- a regiszter és I/O portok

közötti adatátvitel végrehajtható legyen.

- MOV — *adatmozgatás memória és regiszter, regiszter és regiszter között*
- PUSH — *regiszter betöltése verembe*
- POP — *verem tetejének betöltése regiszterbe*
- PUSHF — FLAGS *regiszter mentése verembe*
- POPF — FLAGS *regiszter betöltése veremből*
- IN — *adatolvasás I/O portról*
- OUT — *adatküldés I/O portra.*

Vezérlésátadó utasítások

E körbe tartozó utasítások közül a ciklusképzéshez, a szubrutinokhoz, a feltételes és feltétel nélküli ugráshoz, valamint a programvezérelt megszakításhoz szükséges műveleteket választjuk ki. Ezek:

- LOOP — *CX értékének dekrementálása és ugrás, ha* $CX \neq O$
- CALL — *ugrás szubrutinba* (szubrutin felhívása)
- RET — *visszatérés szubrutinból*
- JMP — *feltétel nélküli ugrás* az utasításban megadott címre
- JZ/JE — *ugrás, ha* $ZF = 1$ / ugrás, ha egyenlő (CMP után)
- JNZ/JNE — *ugrás ha* $ZF = 0$ / ugrás, ha nem egyenlő (CMP után)
- JC/JB — *ugrás, ha* $CF = 1$ / ugrás, ha kisebb
- JNC/JAE — *ugrás ha* $CF = 0$ / ugrás, ha nagyobb egyenlő
- INT — programvezérelt megszakításkérés
- IRET — visszatérés megszakítás-kiszolgáló rutinból.

Bitléptető és forgató utasítások

- SHL *Logikai eltolás balra*
- SHR *Logikai eltolás jobbra*
- RCL *Forgatás balra CF-el*
- RCR *Forgatás jobbra CF-el*

A bitléptető és forgató utasításokat a fejezet második részében bemutatott mintaprogramokban fogjuk felhasználni. Néhány ilyen utasítást azért szerepeltettünk az utasításkészletben, mert ezek nélkül utasításkészletünk nem reprezentálná a valódi processzorokat. Érdemes felhívni a figyelmet arra, hogy bár a szorzás és osztás művelete nincs benne modellparancsunk utasításkészletében, ezek is elvégezhetők ismételt összeadással és a léptető utasítások felhasználásával.

Processzorvezérlő utasítások

- CLC *CF beállítása 0-ra*
- STC *CF beállítása 1-re*
- HLT *Processzor leállítása egy hardver megszakításig*

Miután funkcionálisan rögzítettük modellprocesszorunk utasításkészletét, a következőkben meghatározzuk az **egyes utasítások műveleti kódját**. Ehhez az egyes utasításokban elvégzendő műveletek értékelésével először meghatározzuk, hogy utasításaink közül melyek a kétcímesek, egycímesek illetve 0-címűek. (A processzorvezérlő utasítások, a PUSHF és POPF, valamint a RET és IRET 0-című utasítások).

Soroljuk fel először a kétcímű, majd az egycíműeket, végül a 0 című utasításokat és mindegyikhez rendeljünk hozzá rendre a 4, 6, 8 bites műveleti kódokat (lásd VI.12. sz. ábra).

Utasításkód mnemonik	Műveleti kód binárisan			
	4 bit	2 bit	2 bit	
ADD	0000			
SUB	0001			KÉT-
CMP	0010			
AND	0011			CÍ-
OR	0100			MŰ
XOR	0101			
MOV	0110			UTA-
SHL	0111			SÍ-
SHR	1000			TÁ-
RCL	1001			SOK
RCR	1010			
INC	1011	00		
DEC	1011	01		
NEG	1011	10		
NOT	1011	11		
PUSH	1100	00		EGY-
POP	1100	01		CÍ-
IN	1100	10		MŰ
OUT	1100	11		
LOOP	1101	00		UTA-
CALL	1101	01		SÍ-
JMP	1101	10		TÁ-
JZ/JE	1101	11		SOK
JNZ/JNE	1110	00		
JC/JB	1110	01		
JNC/JAE	1110	10		
INT	1110	11		
RET	1111	00	00	0 CÍ-
IRET	1111	00	01	MŰ
PUSHF	1111	00	10	
POPF	1111	00	11	UTA-
CLC	1111	01	00	SÍ-
STC	1111	01	01	TÁ-
HLT	1111	01	10	SOK

VI.12. sz. ábra
A modellprocesszor utasításkódjai

Vegyük észre, hogy a műveleti kód alapján a processzor egyértelműen el tudja dönteni, hogy kétcímű, egycímű vagy 0 című utasítást olvasott be (lásd VI.13. sz. ábra).

műveleti kód első 4 bitje	$\leq 1010 \rightarrow$	kétcímű utasítás
$1010 <$ műveleti kód első 4 bitje	$\leq 1110 \rightarrow$	egycímű utasítás
műveleti kód első 4 bitje	$= 1111 \rightarrow$	0 című utasítás

VI.13. sz. ábra
Az utasításban szereplő operandusok számának megállapítása a műveleti kód első 4 bitjéből

3. Utasításkészlet gépi kódban, a gépi kódú utasítások végrehajtása

Tulajdonképpen már minden információval rendelkezünk ahhoz, hogy a gépi kódú utasításokat binárisan le tudjuk írni. Nézzünk erre példákat! Az egyes, konkrét utasítások meghatározásához semmi mást nem kell tennünk, csak a VI.5., 11. és 12. sz. ábra adatait felhasználnunk.

1. Példa

Adjuk meg annak az utasításnak a gépi kódját, mely az AX akkumulátorban lévő értéket 5-tel megnöveli. !

Megoldás

- A szükséges utasítás nyilvánvalóan az összeadás lesz. !
- AX-hez egy konkrét számot kell hozzáadnunk, így az 1-es operandus egy regiszter, a 2-es operandus pedig a közvetlen adatcímzésű „5"-ös szám. Az utasítástípus tehát RD.
- A VI.5. sz. ábra táblázata szerint AX kódja: 0000.
- A VI.12. sz. ábra táblázatából kikeressük az összeadás (ADD) műveleti kódját, ez: 0000
- A VI.11. sz. ábra táblázatából kikeressük az RD utasítástípust. A megfelelő sor ezt tartalmazza:

	Első bájt		Második bájt		Harmadik és negyedik bájt
RD	OP	0010	R1	0000	DW

Itt OP az ADD műveleti kódja, tehát 0000
R1 az AX-t jelenti, tehát 000
DW az 5-ös számot jelenti 16 biten binárisan kódolva. Ennek értéke

tehát: | 00000000 | 0000 | 0101 | → 5

Behelyettesítve RD utasításszerkezetébe a keresett utasítás gépi kódját a VI. 14. sz. ábra mutatja.

0000	0010	0000	0000	0000	0000	0000	0101
ADD	RD típus	AX		5			

VI. 14. sz. ábra
Az akkumulátorhoz 5-öt hozzáadó utasítás gépi kódja

2. Példa

Töltsük fel bináris 0-val a CX regisztert! (Ezt a szakmai zsargonban legtöbbször úgy fejezik ki, hogy töröljük CX-et)

Megoldás

Ezt a feladatot az 1. példához hasonlóan a MOV utasítással is megoldhatjuk (ezt gyakorlásként ajánljuk az olvasónak), de „elegánsabb" és gyorsabb megoldás az XOR „kizáró vagy" utasítás alkalmazása.

Az XOR CX, CX utasítás végrehajtása után ugyanis CX tartalma bináris 0 lesz. (Ezt javasoljuk egy példán ellenőrizni az I. fejezetben található igazságtáblázat alapján.)

Az XOR műveleti kódja: 0101
A CX regiszter kódja: 0100

Utasításunk nyilván RR típusú, ezért a keresett gépi kód a következő:

0101	0000	0100	0100
↓	↓	↓	↓
XOR	RR típus	CX	CX

3. Példa

Írjunk egy kis programrészt a következő feladatra:
A memória 255-ik bájtjától kezdődően 6 db 1 bájtos szám található (lásd VI.15. sz. ábra).

			-------	01	02	00	0A	01	02	
0	1	2		255	256	257	258	259	260	← Memóriacímek decimálisan
0000	0001	0002		00FF	0100	0101	0102	0103	0104	← hexadecimálisan

VI. 15. sz. ábra
A 6 db 1 bájtos szám elhelyezkedése a memóriában

Adjuk össze ezt a 6 db számot úgy, hogy az eredmény AX-ben keletkezzen.

1. Megoldás

```
XOR   AX, AX     ; Először AX-et és DX-et törölni kell,
XOR   DX, DX     ; mert nem tudjuk mi a tartalma
MOV   BX, 255    ; BX-be az első bájt címének elhelyezése
MOV   DL, [BX]   ; Vigyél DX alsó 1 bájtjába a
                 ; BX regiszterben található címről 1 bájtot
ADD   AX, DX     ; Add össze AX-et és DX-et (az eredmény
                 ; AX-ben keletkezik)
MOV   BX, 256    ; Az előző 3 utasítást most még 5-ször
MOV   DL, [BX]   ; ismételni kell a fennmaradó 5 bájt össze-
ADD   AX, DX     ; adásához
MOV   BX, 257
MOV   DL, [BX]
ADD   AX, DX
```

```
MOV   BX, 258
MOV   DL, [BX]
ADD   AX, DX
MOV   BX, 259
MOV   DL, [BX]
ADD   AX, DX
MOV   BX, 260
MOv   DL, [BX]
ADD   AX, DX
```

Tegyük fel most a kérdést: hogyan oldanánk meg ezt a feladatot, ha nem 6 db, hanem mondjuk 600 számot kellene összeadni? Nyilván nem szeretnénk a két MOV és az egy ADD utasítást 600-szor megismételni.

2. Megoldás

Vegyük észre, hogy az előző programrészletben
a

```
MOV BX,(?)
MOV DL, [BX]
ADD AX, DX
```

utasításokat úgy ismételtük, hogy csak a (?)-lel jelölt értéket (címet) változtattuk. Ez a cím először 255 volt, azután 256, 257 stb., azaz mindig 1-gyel növekedett. BX értékének 1-gyel való megnövelését viszont megoldhatjuk az INC utasítással.

Ha BX értéke már 255, akkor a következő két utasítássorozat teljesen egyenértékű:

```
MOV BX, 256  }     { INC BX
MOV DL, [BX] }  ≡  { MOV DL, [BX]
ADD AX, DX   }     { ADD AX, DX
```

A jobboldali utasítássorozat már nem tartalmazza a 256, 257 stb. konstansokat. Ezért, ha azt mondhatnánk, hogy ezt a három utasítást ismételd meg hatszor, akkor célhoz jutnánk.

Ezt úgy fogjuk elérni, hogy a CX számlálóregiszter kezdeti értékét 6-tal (a számok darabszámával) feltöltjük, majd minden egyes szám hozzáadását követően CX értékét 1-gyel csökkentjük (DEC utasítás).

Végül egy JNZ (ugrás ha ZF = 0, azaz CX ≠ 0) utasítással „visszaugrunk" az összeadást addig ismételni, amíg CX értéke 0 nem lesz. Ekkor már ZF = 1-re lesz állítva, tehát nem fog bekövetkezni visszaugrás.

Ezt figyelembe véve a feladatot megoldó programrészletünk a következő lesz:

```
        XOR   AX, AX
        XOR   DX, DX
        MOV   BX, 255
        MOV   CX, 6
ISMET:  MOV   DL, [BX]
        ADD   AX, DX
        INC   BX        ; BX értékének növelése 1-gyel, hogy a
                        ; következő bájtot címezze meg
        DEC   CX        ; CX értékének csökkentése 1-gyel
        JNZ   ISMET     ; ugrás, ha CS ≠ 0
```

E programrészletben az „ISMET:" az Assembly programnyelvben alkalmazható ún. **szimbolikus címke**, melyet a fordítóprogram a MOV DL, [BX] utasítás memóriabeli kezdőcímének fog értelmezni.

A következő táblázatban feltüntetjük, hogy az előző programrészlet végrehajtása során az utasítások befejezésekor az AX, BX, CX, DX regiszterek tartalma milyen hexadecimális értéket vesz fel (A 6 db 1 bájtos szám a VI.15. sz. ábra szerinti értékeket tartalmazza.).

Javasoljuk az olvasónak, hogy egy ilyen üres táblázatot először a programutasítások alapján próbáljon meg kitölteni, majd az eredményt a VI.16. sz. ábra alapján ellenőrizze.

	Utasítás sorszáma	AX	BX	CX	DX
	1	0000	?	?	?
	2	0000	?	?	0000
	3	0000	00FF	?	0000
	4	0000	00FF	0006	0000
1 ÁTFUTÁS	5	0000	00FF	0006	0001
	6	0001	00FF	0006	0001
	7	0001	0100	0006	0001
	8	0001	0100	0005	0001
	9	0001	0100	0005	0001
2 ÁTFUTÁS	5	0001	0100	0005	0002
	6	0003	0100	0005	0002
	7	0003	0101	0005	0002
	8	0003	0101	0004	0002
	9	0003	0101	0004	0002
3 ÁTFUTÁS	5	0003	0101	0004	0000
	6	0003	0101	0004	0000
	7	0003	0102	0004	0000
	8	0003	0102	0003	0000
	9	0003	0102	0003	0000
4 ÁTFUTÁS	5	0003	0102	0003	000A
	6	000D	0102	0003	000A
	7	000D	0103	0003	000A
	8	000D	0103	0002	000A
	9	000D	0103	0002	000A
5 ÁTFUTÁS	5	000D	0103	0002	0001
	6	000E	0103	0002	0001
	7	000E	0104	0002	0001
	8	000E	0104	0001	0001
	9	000E	0104	0001	0001
6 ÁTFUTÁS	5	000E	0104	0001	0002
	6	0010	0104	0001	0002
	7	0010	0105	0001	0002
	8	0010	0105	0000	0002
	9	0010	0105	0000	0002

Jelölés

(szürke mező) =

A megfelelő utasítás végrehajtása során az eredményadatok helye

VI.16. sz. ábra
A regisztertartalmak változása programrészletünk végrehajtása során

Látható, hogy programrészletünk végrehajtása során az 5, 6, 7, 8, 9 sz. utasítások 6-szor kerülnek végrehajtásra.

Ha egy utasítássorozatot egymás után többször, változatlan formában, de általában változó adatokkal hajtunk végre a program futtatása során, akkor ezt **ciklus**nak nevezzük.

Az egyik gyakori programhiba az úgynevezett „örök ciklus". Az előző programpéldánkban ez az eset állt volna elő, ha a 4 sorszámú MOV CX, 6 utasítást az utasítássorból „kifelejtjük" és a CX regiszter tartalma a ciklusba való belépéskor 0000 vagy egy negatív szám. Ebben az esetben CX értéke a dekrementálások hatására negatív lesz és ezáltal a ZF flag sohasem fogja felvenni az 1 értéket. Emiatt a program végrehajtásakor az 5, 6, 7, 8, 9 utasításokat a számítógép „végtelen sokszor" megismétli.

Az előző programrészletet felhasználjuk arra, hogy a gépi kódú programokat is bemutassuk, és kis betekintést nyerjünk az Assembly nyelvű programok fordítóprogramjának működésébe.

Tételezzük fel, hogy programrészletünk a memóriában közvetlenül a 6 db 1 bájtos szám után helyezkedik el. Ekkor kezdőcíme decimálisan 261, hexadecimálisan pedig 0105. A korábban már megismert módon állítsuk elő a 9 db utasítás gépi kódját (VI.17. sz. ábra).

		1 bájt		1 bájt		1 bájt		1 bájt	
	XOR AX, AX	0101	0000	0000	0000				
	XOR DX, DX	0101	0000	0110	0110				
	MOV BX, 255	0110	0010	0010	0000	0000	0000	0000	1111
	MOV CX, 6	0110	0010	0100	0000	0000	0000	0000	0110
ISMET:	MOV DL, [BX]	0110	0001	0111	0010				
	ADD AX, DX	0000	0000	0000	0110				
	INC BX	1011	0000	0000	0010				
	DEC CX	1011	0100	0000	0100				
	JNZ ISMET	1110	0010	0000	0001	0001	0001		

VI.17. sz. ábra
Programrészletünk gépi kódban

A VI.17. sz. ábra táblázatát a VI.5., 11. és 12. sz. ábrák táblázatai alapján állítottuk össze. Ennek során a lényegét tekintve a **fordítóprogramot szimuláltuk**:

- A szimbolikus műveleti kód mnemonikok (pl. XOR, MOV stb.) alapján kikerestük a műveleti kód táblázatból a műveleti kód bináris értékét.

- „Felismertük" a regiszterek szimbolikus nevét (AX, BX, stb.) ez alapján a regiszterkódtáblázatból kikerestük a regiszterek bináris kódját.
- Az utasításokban található regiszter darabszám, konstansok, az indirekt címzést jelölő [] jel alapján meghatároztuk, hogy az utasításoknak milyen az utasításszerkezete. Ezt követően az utasításszerkezetre vonatkozó táblázatból kikerestük az utasítás formai felépítését.

! Ha mindezt megtettük, ezt követően már csak az utasításokban lévő konstansokat kellett binárisra átkonvertálni és a szimbolikus címeknek (ISMET:) a bináris értékét meghatározni. Ezt követően kitölthettük a VI.17. sz. ábra táblázatát gépi kódban.

A **szimbolikus címek bináris értékének meghatározása** még további magyarázatot igényel.

Mint már említettük az Assemly konvenciók szerint az „ISMET:" szimbolikus cím a MOV DL, [BX] utasítás gépi kódjának tárolóbeli címét jelenti. Ahhoz, hogy a JNZ utasításban lévő ugrási cím (mely szimbolikusan ISMET:) bináris értékét meghatározzuk egy rövid számításra van szükség.

Ennek során először az ISMET: címkével jelölt MOV utasítás gépi kódjának a programrészlet elejétől való távolságát határozzuk meg bájtokban számolva.

A VI.17. sz. ábra táblázata szerint az

ISMET: MOV DL, [BX]

utasítást 2 darab 2 bájtos (2 db. XOR) és 2 darab 4 bájtos (2 db. MOV) utasítás előzi meg. Ez összesen

$$2 \cdot 2 + 2 \cdot 4 = 12$$

bájtot jelent. Ez azt jelenti, hogy az ISMET: szimbolikus címnek megfelelő cím a programrészlet kezdőcímétől 12 bájtra található. Mivel tudjuk, hogy programrészletünk kezdő címe a memóriában 261, ezért ISMET: címe:

261 + 12 = 273 vagy hexadecimálisan és

binárisan: 0111 = 0000 0001 0001 0001

! Példánk alapján világos, hogy az Assembly nyelvű programokból egy jól felépített szabály és eljárás rendszer alapján előállíthatjuk az utasítások gépi kódját. Erre az algoritmusra tehát programot is írhatunk, amit **Assembler**-nek nevezünk.

Az Assembler tehát az a fordítóprogram, mely az Assembly nyelven megadott programutasításokból gépi kódú utasításokat állít elő.

4. Az utasításkészlet

Az elkövetkezendőkben sorra vesszük az utasításkészlethez tartozó utasításokat és megadjuk

- az egyes utasítások formátumát (szintaxisát) Assembly nyelven;
- az adott utasítás esetén alkalmazható utasítástípusokat;
- röviden bemutatjuk az utasítások végrehajtását, és
- egyes esetekben utalunk az adott utasítás jellemző felhasználására a programozásban, melyet példákkal is illusztrálunk.

Megjegyezzük, hogy az elkövetkezendőkben bemutatott utasításkészlet az IBM PC Assembly egy jelentősen szűkített változata. Ez többek között azt is jelenti, hogy az input/output utasítástól eltekintve a tárgyalt utasítások és szintaxis az IBM PC Assembly-ben is változtatás nélkül alkalmazhatók.

4.1. Aritmetikai utasítások

Összeadás (Addition Integers)

Szintaxis: ADD cél, forrás

Megengedett utasítástípusok:

- regiszterhez regisztert — RR
- regiszterhez memóriát — RI, RB
- regiszterhez konstanst — RD
- memóriához regisztert — IR, BR

Az utasítás hatása: két egész szám, mint operandus összeadása.

Az utasítás működése: Az ADD utasítás két operandus összeadását végzi el, az eredményt a céloperandus helyére írja.
(forrás + cél) → cél

Példák:		
	Regisztert regiszterhez	ADD AX, BX
	Regisztert memóriához	ADD [BX], AX
	Konstanst regiszterhez	ADD AX, 1999
	Memóriát regiszterhez	ADD BX, [BX+SI]

Kivonás (Subtraction Integers)

Szintaxis: SUB cél, forrás

Megengedett utasítástípusok:

- regiszterből regisztert — RR
- regiszterből memóriát — RI, RB
- regiszterből konstanst — RD

- memóriából regisztert IR, BR

Az utasítás hatása: két egész szám kivonása.

Az utasítás működése: A SUB utasítás a forrásoperandust kivonja a céloperandusból, majd az eredményt a céloperandus helyére írja. SF és ZF beállításra kerül.
(forrás – cél) → cél

Példák:	Regiszterből regisztert	SUB AX, BX
	Regiszterből konstanst	SUB CX, 5
	Memóriából regisztert	SUB [BX + SI], CX
	Regiszterből memóriát	SUB AX, [BX]

Egész számok összehasonlítása (Compare Integers)

Szintaxis: CMP cél, forrás

Megengedett utasítástípusok:

- regisztert regiszterrel RR
- regisztert memóriával RI, RB
- regisztert konstanssal RD
- memóriát regiszterrel IR, BR

Az utasítás hatása: Flag-ek beállítása.

Az utasítás működése: A CMP utasítás kivonja a forrásoperandust a céloperandusból. Az operandusok tartalma változatlan marad, de a kivonás eredményétől függően a ZF és az SF flag beállításra kerül.

Példák:	Regisztert regiszterrel	CMP AL, BL
	Regisztert memóriával	CMP DX, [SI]
	Regisztert konstanssal	CMP BL, 05H

Utolsó példánkban a „H” az Assembly szintaxis szerint hexadecimálisan értelmezendő számot jelöli.

10 → 00001010
10H → 00010000

Inkrementálás (Increment)

Szintaxis: INC cél

Megengedett utasítástípus:

regiszter R

Az utasítás hatása: A célregiszter inkrementálása.

Az utasítás működése: Az INC utasítás az operandus értékéhez 1-et hozzáad.
(operandus + 1) → operandus

Példa INC CX

Dekrementálás (Decrement)

Szintaxis: DEC cél
Megengedett utasítástípus:
regiszter R
Az utasítás hatása: A célregiszter dekrementálása.
Az utasítás működése: A DEC utasítás az operandus értéket 1-gyel csökkenti.
(operandus – 1) → operandus
Példa: DEC SI

Kettes komplemens képzése (Negate Integer)

Szintaxis: NEG cél
Megengedett utasítástípus:
regiszter R
Az utasítás hatása: Egy bájt vagy szó kettes komplemensének képzése.
Az utasítás működése: A NEG utasítás kivonja az operandust 0-ból és ennek eredményéhez 1-et ad hozzá. Az így kapott számot beírja az operandus helyére.
Példa a programozásban való felhasználásra: a számok abszolútértékének előállítása programmal

```
        OR AX, AX     ; Ennek hatására AX értéke nem
                      ; változik, viszont az SF flag-et beállítja
        JNS POZ       ; ugrás ha pozitív
        NEG AX
POZ:
```

Az utasítássor hatására AX az eredetileg AX-ben lévő szám abszolutértékét tartalmazza.

4.2. Logikai utasítások

Logikai ÉS művelet (AND)

Szintaxis: AND cél, forrás
Megengedett utasítástípusok:

- regisztert regiszterrel RR
- regisztert memóriával RI, RB
- regisztert konstanssal RD
- memóriát regiszterrel IR, BR

Az utasítás hatása: A két operandus között logikai „ÉS" művelet végrehajtása bitről-bitre.

Az utasítás működése: Az eredmény azokban a bitpozíciókban lesz 1-es, ahol mindkét operandusban 1-es szerepelt, egyébként 0. A művelet eredménye a céloperandus helyére kerül beírásra. ZF beállításra kerül.

Példa: AND AX, BX
AND DL, 11001100B

Itt az utasításban szereplő „B" Assembly szintaxis szerint 1 bináris szám megadását jelöli

Ha	DL tartalma:	10101010
	konstans:	11001100
	DL tartalma az utasítás végrehajtása után	10001000

Logikai megengedő VAGY művelet (OR)

Szintaxis: OR cél, forrás

Megengedett utasítástípusok:

- regisztert regiszterrel RR
- regisztert memóriával RI, RB
- regisztert konstanssal RD
- memóriát regiszterrel IR, BR

Az utasítás hatása: A két operandus között logikai „VAGY" művelet végrehajtása bitről-bitre.

Az utasítás működése: Az eredmény azokban a bitpozíciókban lesz 0, ahol mindkét operandusban 0 található. Minden más esetben az eredmény 1. A művelet eredménye a céloperandus helyére kerül beírásra és ZF beállításra kerül.

Példa: OR CX, DI
OR AI, 804

Logikai kizáró VAGY (Exclusive OR)

Szintaxis: XOR cél, forrás

Megengedett utasítástípusok:

- regisztert regiszterrel RR
- regisztert memóriával RI, RB
- regisztert konstanssal RD
- memóriát regiszterrel IR, BR

Az utasítás hatása: A két operandus között logikai kizáró „VAGY" művelet végrehajtása bitről-bitre

Az utasítás működése: Az eredmény azokban a bitpozíciókban lesz 1-es, ahol az operandusok bitjei eltérőek. A művelet eredménye a céloperandus helyére kerül beírásra és ZF beállításra kerül.

Példa: XOR AX, AX ; Regiszter törlése

Logikai NEM művelet (Complement)

Szintaxis: NOT cél

Megengedett utasítástípus:

regiszter R

Az utasítás hatása: Az operandus minden bitjét negálja (egyes komplemens).

Az utasítás működése: A NOT utasítás az operandus minden bitjét ellenkezőjére állítja be és ezt az operandus címén helyezi el. Az utasítás végrehajtása a flag-eket nem befolyásolja.

Példa: NOT CX

4.3. Adatmozgató utasítások

Adatmozgatás (Move Data)

Szintaxis: MOV cél, forrás

Megengedett utasítástípusok:

- regiszterből regiszterbe RR
- regiszterből memóriába IR, BR
- memóriából regiszterbe RI, RB
- konstans regiszterbe RD

Az utasítás hatása: A forrásoperandus átmásolása a céloperandusba.

Az utasítás működése: A céloperandus felveszi a forrásoperandus értékét, a forrásoperandus változatlan marad (ha a kettő nem azonos). Az utasítás végrehajtása a flag-eket nem állítja be.

Példák

```
MOV CL, 1      ; CL értéke 00000001 lesz
MOV AX, BX     ; BX átmásolása AX-be
MOV BX, [BX]   ; BX tartalma a művelet után a korábban
               ; BX-el címzett 2 bájt lesz
```

Veremműveletek (Stack Instructions)
Verem tetejéről szó betöltése regiszterbe

Szintaxis: POP cél
POPF

Megengedett utasítástípusok:
regiszter R

Az utasítás hatása:

POP: A verem tetején lévő szót átviszi a céloperandusba, majd a veremmutató SP regiszter értékét 2-vel növeli.

POPF: A verem tetején lévő szót átviszi a FLAGS regiszterbe, majd a veremmutató SP regiszter értékét 2-vel növeli.

Az utasítás működése:

POP: Az SP regiszter által megcímzett 2 bájt átkerül a célopeandusba, majd SP értéke 2-vel növekszik. A flag-ek értékét nem változtatja.

POPF: Az SP regiszter által megcímzett 2 bájt átkerül FLAGS regiszterbe, majd SP értéke 2-vel növekszik. Az utasítás a flag-ek értékét nem változtatja.

Példák: POP CX
POPF

Szó ráhelyezése a verem tetejére

Szintaxis: PUSH forrás
PUSF

Megengedett utasítástípusok:
regiszter R

Az utasítás hatása:

PUSH: A forrásoperandusban megcímzett szót ráhelyezi a verem tetejére.

PUSHF: A FLAGS regiszter tartalmát ráhelyezi a verem tetejére.

Az utasítás működése: A PUSH és a PUSF utasítás az SP veremmutató regiszter értékét 2-vel csökkenti, majd ezt követően a forrásoperandus regiszterben található értéket illetve a FLAGS regiszter tartalmát beírja az SP által címzett memóriaterületre. Az utasítás a flag-ek értékét nem változtatja.

Példák:
```
PUSHF
PUSH BX
```

Példa a PUSH, PUSHF, POP, POPF utasítások alkalmazására szubrutinokban.

Feladat: Egy szubrutin paraméterét az AX regiszterben kapja meg és az eredményt szintén AX-ben adja vissza. Az algoritmusban szükségünk van két darab munka- és egy indexregiszterre. Hogyan kell megírni a szubrutint, ha azt akarjuk, hogy a visszaugrást követően csak AX tartalma változzon meg?

A feladatnak megfelelő szubrutin felépítése:
```
SUB:  PUSHF       ; FLAGS tartalmát is meg kell őrizni.
      PUSH BX
      PUSH CX
      .
      .    A szubrutin ezekben az utasításokban
      .    szabadon változtathatja AX, BX, CX
      .    és FLAGS tartalmát.
      .
      POP CX
      POP BX
      POPF
      RET         ; visszaugrás
```

A szubrutinban tehát először kimentjük a FLAGS, BX, CX regiszterek tartalmát a verembe, majd visszaugrás előtt **fordított sorrendben(!!)** visszatöltjük.

Bájt bevitele I/O portról (Input from I/O port)

Szintaxis: IN 0
IN 1
IN 3

Az utasítás hatása:

IN 0 = 1 bájtot átvisz a 0-s I/O port pufferregiszteréből az AL-be

IN 1 = 1 bájtot átvisz a 0-s I/O port állapotregiszterből az AL-be

IN 3 = 1 bájtot átvisz az 1-es I/O port állapotregiszteréből az AL-be

Bájt kivitele az 1-es I/O port pufferébe (Output to Port)

Szintaxis: OUT 2

Az utasítás hatása: 1 bájtot átvisz AL-ből az 1-es I/O port pufferregiszterébe.

4.4. Vezérlésátadó utasítások

Ciklusszervezés a CX regiszter használatával (Decrement CX and Branch)

Szintaxis: LOOP ugrási cél

Megengedett utasítástípus: D, azaz konstans (szimbolikus cím)

Az utasítás hatása: CX dekrementálása és ugrás ha CX ≠ 0.

Az utasítás működése: A LOOP utasítás először a CX regiszter értékét csökkenti 1-gyel, majd ezt követően megvizsgálja, hogy CX = 0 feltétel teljesül-e. Ha nem, akkor a vezérlés átadásra kerül a LOOP utasításban lévő ugrási célcímre. Ha a CX = 0, akkor a program a LOOP utasítást követő utasítással folytatódik. Az ugrási célcím a LOOP utasítás címétől maximum ±128 bájttal térhet el. (D értéke 1 bájt)

Az utóbbi feltétel vizsgálatával – mivel utasításaink maximum 4 bájtosak – csak akkor kell foglalkoznunk, ha a LOOP utasítással szervezett ciklusban 30-nál több utasítás található. Ez a gyakorlatban nagyon ritkán előforduló eset.

Példafeladat a LOOP utasítással szervezett ciklusra: Számítsuk ki 100 db 2 bájtos szám összegét, mely számok az SI regiszterrel címzett tömbben találhatók!

Megoldás:

```
        XOR AX, AX      ; AX törlése
        MOV CX, 100
UJRA:   ADD AX, [SI]
        INC SI          ; Az SI-ben lévő cím megnövelése
        INC SI          ; 2-vel, mivel számaink 2 bájtosak
        LOOP UJRA
```

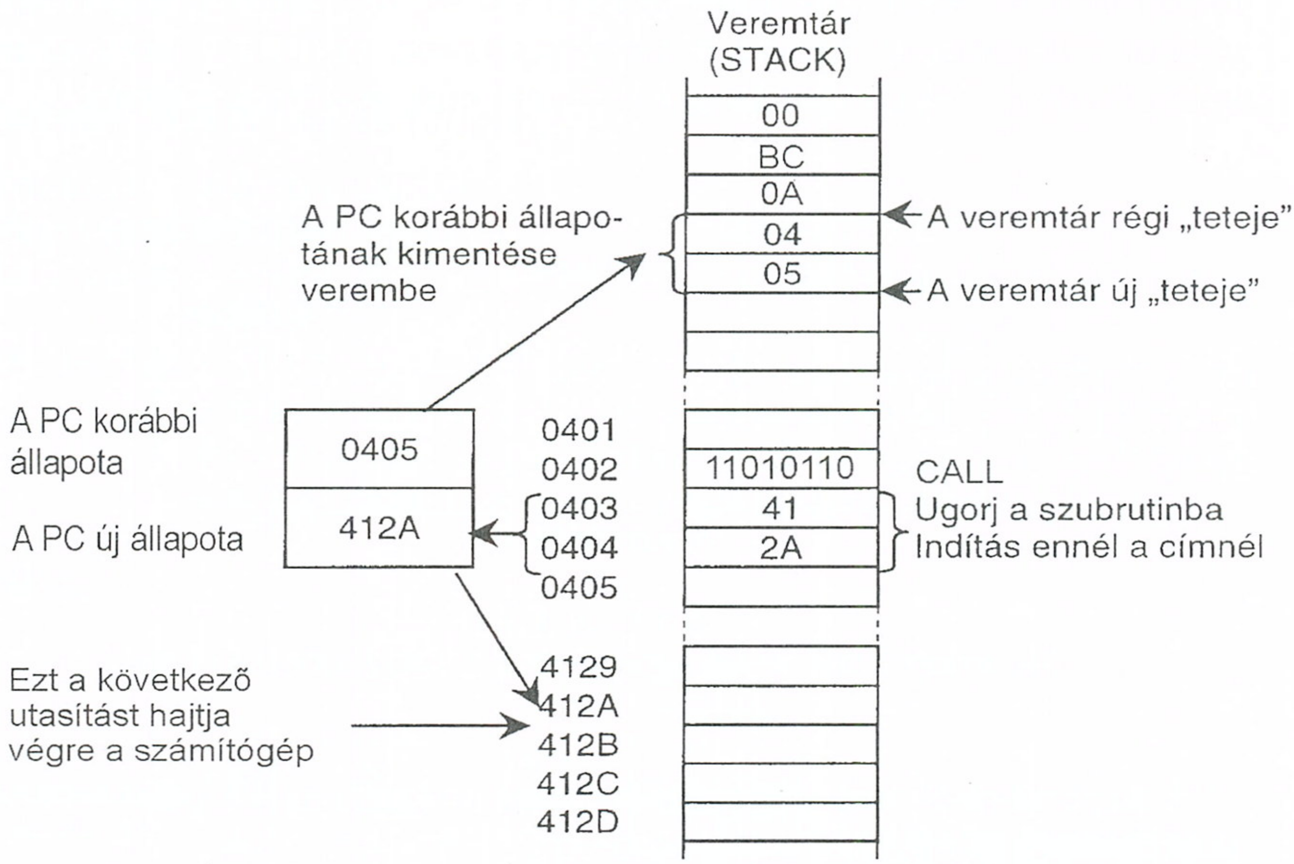
Szubrutinhívás (CALL)

Szintaxis: CALL szubrutincím

Megengedett utasítástípus: D

Az utasítás hatása: A CALL utasítás a következő utasítás címét el menti a verembe, majd a vezérlést átadja a szubrutincímre.

Az utasítás működése: A CALL utasítás először az SP veremmutató regiszter tartalmát 2-vel csökkenti, majd az SP által meghatározott címre kimenti az utasításszámláló IP regiszter tartalmát. Ezt követően az IP regiszter feltöltésre kerül a szubrutincímmel és így a program a meghívott szubrutin első utasításával folytatódik. Az utasítás a flag-eket nem befolyásolja.



VI.18. sz. ábra
A CALL utasítás ugrás a szubrutinba, a veremtárba való egyidejű mentéssel

Visszatérés szubrutinból (Return from Subroutine)

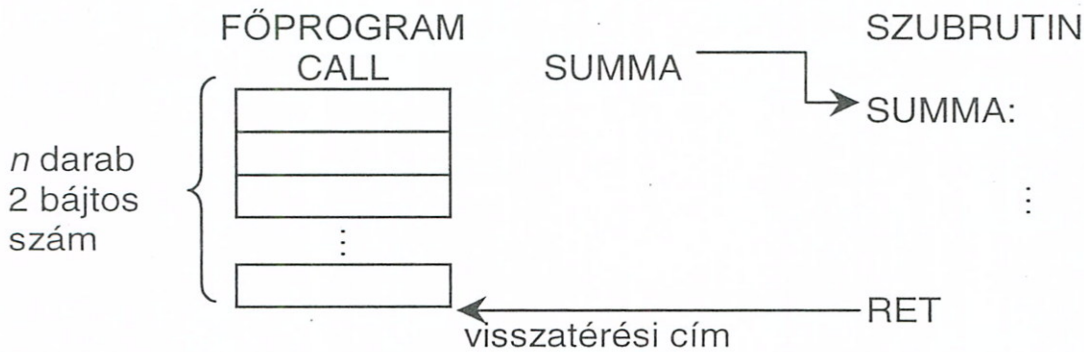
Szintaxis: RET

Az utasítás hatása: Visszaadja a vezérlést a szubrutinhívó CALL utasítást követő utasításnak.

Az utasítás működése: A RET utasítás az SP veremmutató regiszter által megcímzett 2 bájtot áttölti az IP utasításszámlálóba, majd 2-vel megnöveli SP tartalmát. Ezt követően a program végrehajtása az IP-ben lévő címen folytatódik. Az utasítás a flag-ek értékét nem változtatja meg.

Példafeladat

Írjunk szubrutin, mely összead *n* darab 2 bájtos számot és az eredményt az AX regiszterbe helyezi el. Az *n* értékét a szubrutinhíváskor CX tartalmazza. Az *n* darab szám az VI.19. sz. ábra szerint közvetlenül a CALL utasítás után helyezkedik el a memóriában.



VI.19. sz. ábra
Szubrutinparaméterek elhelyezése a CALL utasítás után

A szubrutinból való visszatéréskor az *n* darab szám után található utasítással kell folytatni a főprogramot.

Megoldás:

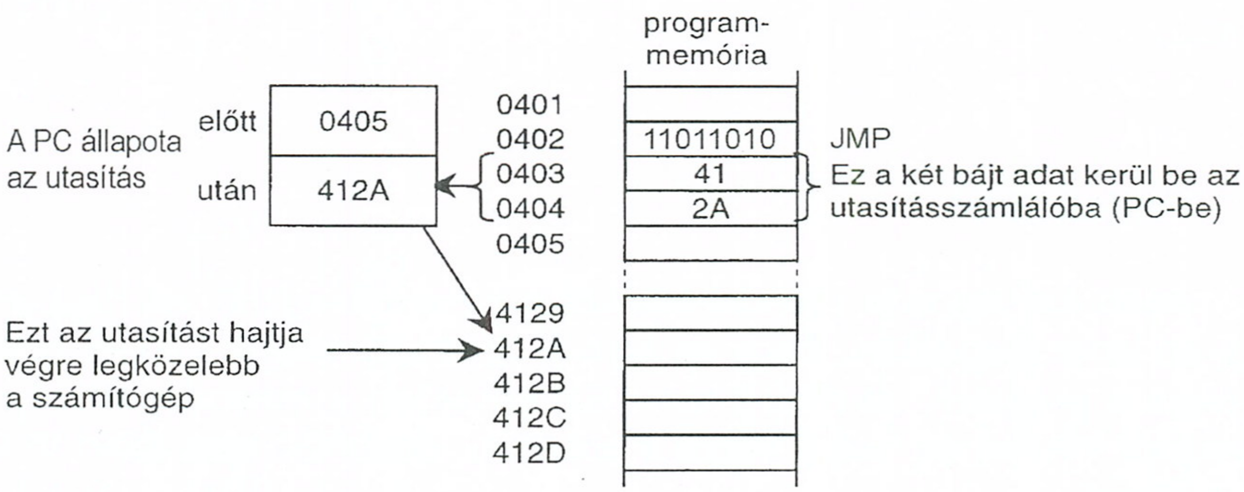
Mivel a CALL utasítás az utasítást követő címet menti el a verembe, a 2 bájtos számokból álló tömb ez alapján címezhető.
Arra kell vigyázni, hogy a RET utasítás előtt a helyes visszatérési cím legyen a verem tetején.

```
SUMMA:  XOR AX, AX     ; AX törlése
        POP SI         ; A CALL utasítás utáni cím betöl-
                         tése a veremből az SI regiszter-
                         be
ISMET:  ADD AX, [SI]
        INC SI
        INC SI
        LOOP ISMET     ; CX-ben van a darabszám
        PUSH SI        ; SI a visszatérési címre mutat,
        RET            ; mivel már végigléptettük a töm-
                       ; bön
```

Feltétel nélküli ugrás (Jump)

Szintaxis: JMP ugrási cél
Megengedett utasítástípus: D
Az utasítás hatása: A programvégrehajtás az ugrási cél címen folytatódik.
Az utasítás működése: A JMP utasítás végrehajtásánál az ugrási célcím értéke betöltődik az utasításszámláló IP regiszterbe, és a program végrehajtása az IP-ben lévő címen folytatódik. Az utasítás a flag-ek értékét nem változtatja meg.

Példa:



VI.20. sz. ábra
A JMP feltétel nélküli ugróutasítás

Feltételes ugrások (Jump if Condition)

! *Szintaxis:* J (tesztfeltétel) ugrási cím

Az utasítás hatása:

Z/E	ugrás, ha ZF = 1 illetve ugrás, ha egyenlő (ekkor egy CMP utasítás előzi meg)
NZ/NE	ugrás, ha ZF = 0 illetve ugrás, ha nem egyenlő (ekkor egy CMP utasítás előzi meg)
C/B	ugrás, ha CF = 1 illetve ugrás, ha kisebb (ekkor egy CMP utasítás előzi meg)
NC/AE	ugrás, ha CF = 0 illetve ugrás, ha nagyobb vagy egyenlő (előtte CMP)

! *Az utasítás működése:* Ha a tesztfeltételek teljesülnek, akkor a feltételes ugróutasítás az ugrási címnek adja át a vezérlést.

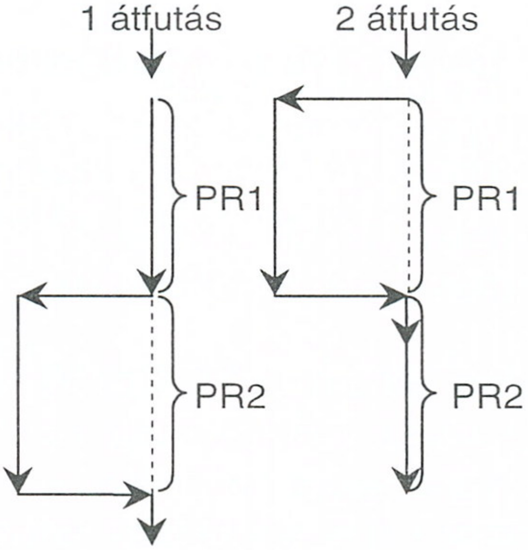
Példák:

```
        MOV AX, 1
        MOV BX, 2
        CMP AX, BX
  ┌──── JB  KIS         ; Mivel AX értéke kisebb mint BX
  ↓     .               ; értéke, a programvégrehajtás
KIS:                    ; a KIS című utasítással folytatódik

        XOR AX, AX
  ┌──── JZ NUL          ; Elugrik mivel AX = 0
  ↓     .
NUL:
```

Példafeladat

Adott két egymás után következő programrész PR1 és PR2. Hogyan lehet megoldani, hogy az első programátfutásnál PR1-et hajtsa végre a program, de PR2-t nem. Ezt követően a második átfutásnál viszont PR1-el ne hajtsa végre a program, de PR2-t igen (lásd VI.21. sz. ábra).



VI.21. sz. ábra
Kapcsolóállítással megoldandó programozási feladat

A feladat megoldásához egy kapcsolót fogunk felhasználni, mégpedig a DL regisztert. Ha DL = 01h, akkor PR1 végrehajtásra kerül, de PR2 nem. Ha DL = 00h, akkor PR1 nem kerül végrehajtásra, de PR2 igen.

```
        XOR   DX, DX
        MOV   DL, 1
HAS:    CMP   DX, 0
        JZ    PR2
PR1:    .
        .

        CMP   DX, 1
        JZ    UGR
PR2:    .
        .
        .
        JMP   TOV
UGR:    MOV   DL, 0
        JMP   HAS
TOV:
```

Programvezérelt megszakításkérés, visszatérés megszakításkiszolgáló rutinból

Szintaxis: INT konstans $0 \leq$ konstans ≤ 255
IRET

Az utasítás hatása:

INT: Megszakítási rutin meghívása
IRET: Megszakítási rutinból visszatérés

Az utasítások működése:

INT: Az utasítás először a FLAGS és az IP regiszter tartalmát a verembe írja, majd az utasításban lévő konstanst indexként használva a memóriában tárolt megszakítási vektortáblázatból kiolvassa a megfelelő megszakítás-kiszolgáló rutin kezdőcímét és átadja annak a vezérlést.

IRET: Az utasítás a veremből feltölti a FLAGS és az IP regisztereket és ezt követően a megszakított program folytatódik.

Ha a számítógépünk DOS operációs rendszerrel működik, akkor az INT utasítással a BIOS (Basic Input Output System) és a DOS előre leprogramozott rutinjait lehet felhívni. Ezeket tulajdonképpen az operációs rendszer szubrutinjainak tekinthetjük.

A megszakítás-kiszolgáló rutinok számára a paramétereket a megfelelő regiszterek feltöltésével tudjuk átadni. Erre egy példa:

Mozgassuk el az egér kurzort a képernyő (10,5) koordinátájú pontjába. Ezt a következő programrészlettel tudjuk végrehajtani:

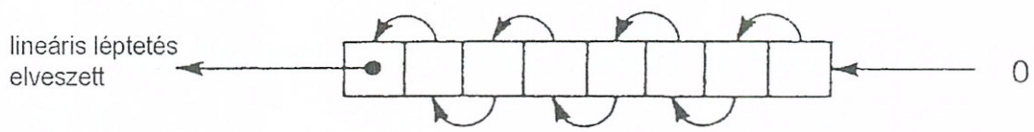
```
MOV AX, 04H    ; ezzel jelezzük, hogy az egér
               ; mozgatása a feladat
MOV CX, 10     ; x koordináta beállítása
MOV DX, 5      ; y koordináta beállítása
INT 33H
```

(Ezeket a paraméterértékeket a megfelelő kézikönyvekből kereshetjük ki.)

4.5. Bitléptető és forgató utasítások

Léptető és körbeléptető műveletekről általában

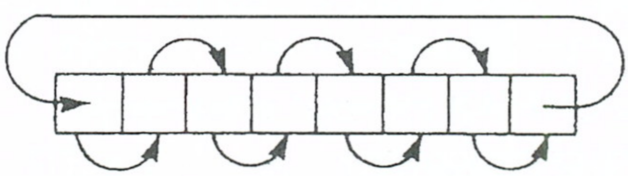
A léptető (SHIFT) és körbeléptető vagy forgató (ROTATE) műveletek nagyon fontosak a szorzási és osztási algoritmusoknál, valamint a különböző számítási műveleteknél.



VI.22. ábra
Lineáris léptetőregiszter

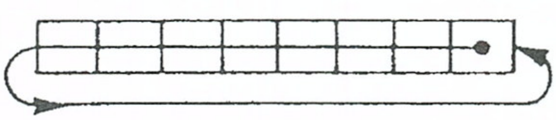
A fent bemutatott **egyszerű léptetés** balra 1 bittel minden egyes bitet átvisz a balra lévő legközelebbi bithelyre és mivel a legnagyobb helyi értékű bittől balra már nincs több bit, így ez elvész. Nem található a legkisebb helyi értékű bittől jobbra sem bithely, így a 0-t visszük be a legkisebb helyi értékű bithelyre.

A **körbeléptető művelet** „önmagába visszatérő” bitátvitel, így a művelet végrehajtása után a legnagyobb és a legkisebb helyi értékű bitek egymással szomszédosak lesznek:

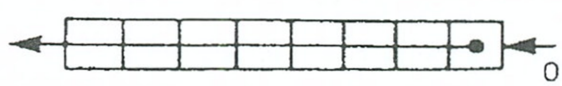


VI.23. ábra
Regiszter forgatása jobbra

A léptető és körbeléptető műveleteknek számos változata lehetséges, így lehet balra illetve jobbra léptetni, illetve körbeléptetni:



VI.24. sz. ábra
Bal irányú körbeléptetés



VI.25. sz. ábra
Lineáris balra léptetés

Léptetés és körbeléptetés jobbra:



VI 26. sz. ábra
Körbeléptetés jobbra



VI. 27. sz. ábra
Lineáris jobbra léptetés

Logikai eltolás jobbra, illetve balra (Shift Right/Left)

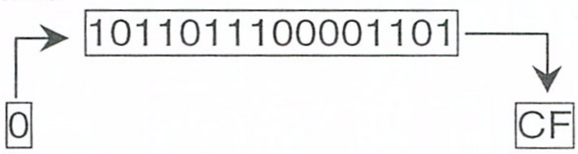
Szintaxis: SHR cél (eltolás jobbra)
SHL cél (eltolás balra)

Megengedett utasítástípus:
regiszter R

Az utasítás hatása: Az utasításban szereplő célregiszter bitjei jobbra, illetve balra lépnek 1-bitet.

Az utasítás működése: Az utasításban megadott regiszterben lévő bitsorozat jobbra illetve balra lép 1-bitet, a regiszterből kilépő bit értéke beállítja CF-et, baloldalon illetve jobboldalon 1 db 0-s bit lép be a regiszterbe. Az eredményt a céloperandus tárolja.

Példa: ha AX tartalma



akkor SHR AX művelet eredménye:

0101101110000110 és CF = 1

Forgatás jobbra, illetve balra CF-el (Rotate Through Carry Right/Left)

Szintaxis: RCR cél (forgatás jobbra)
RCL cél (forgatás balra)

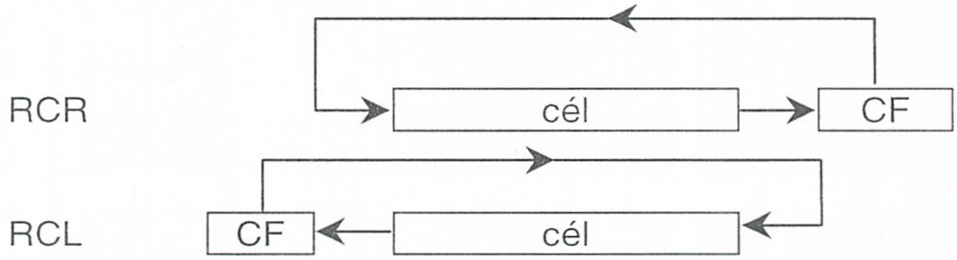
Megengedett utasítástípus:
regiszter R

Az utasítás hatása: A célregiszteren bitforgatási művelet jobbra illetve balra.

Az utasítás működése: Az RCR utasítás a CF flag értékét helyezi a célregiszter legnagyobb helyi értékű bitjébe, a legkisebb helyi értékű bitet pedig átviszi a CF flag-be.

Az RCL utasítás a CF flag bitjét a célregiszter legkisebb helyi értékű bitjébe helyezi át és a legnagyobb helyi értékű bittel beállítja a CF flag-et.

Emellett az utasítás hatására a célregiszter bitjei 1 pozíciót lépnek jobbra illetve balra.



VI.28. sz. ábra
Az RCR és RCL művelet végrehajtása

Példák: RCR BL
RCL AX

4.6. Processzorvezérlő utasítások

Ebbe a csoportba tartozó utasítások közvetlenül befolyásolják a processzor működését, illetve a FLAGS regiszter egyes bitjeinek beállítására szolgálnak.

Leállítás (Halt)

Szintaxis: HLT
Megengedett utasítástípus: 0 címes
Az utasítás hatása: Az utasítások végrehajtásának leállítása.
Az utasítás működése: A HLT utasítás felfüggeszti a program végrehajtását. Ebből az állapotból való újraindítás csak külső hardver megszakítással vagy törléssel (RESET gomb) lehetséges. Ha megszakítással indítjuk újra a programot, akkor ez a HLT utasítást követő utasítással folytatódik.

Az Intel processzorok többfeladatos, úgynevezett védett üzemmódjában a HLT utasítás privilegizált utasítás, ami azt jelenti, hogy a felhasználói programok számára használata nem megengedett.

A Carry flag beállítása

CLC = Clear Carrí Flag, a carry flag törlése
STC = Set Carry Flag, a carry flag beállítása
Szintaxis: CLC
STC
Megengedett utasítástípus: 0 címes
Az utasítás hatása, működése: A CF flag értékét a CLC utasítás 0-ra, az STC utasítás pedig 1-re állítja be.

5. Az utasításkészlet felhasználása a programozásban

Ebben a pontban egy konkrét, viszonylag komplex programozási feladat megoldásával bemutatjuk az utasításkészlet felhasználását a programozás gyakorlatában.

Példáink kiterjednek a programozás tipikus módszereire, így például a következőkre:

- ciklusok szervezése,
- szubrutinok írása,
- I/O műveletek programozása,
- bitsorozatok tesztelése logikai utasításokkal,
- tömbök kezelése.

5.1. A programozási feladat

Adott egy számítógépes rendszer, melynek maximum 256 felhasználója lehet. A felhasználókat azonosítószámokkal látjuk el, ezeket 0–255 számtartományból választjuk ki. A felhasználói azonosító a számítógépen belül egy 1 bájtos bináris szám.

A felhasználók két könyvtárrendszerrel végezhetnek műveleteket, az egyik könyvtár adatokat, a másik programokat tartalmaz. Ezek neve legyen ADAT, illetve PROG. A felhasználók a könyvtárak használata során a következő műveleteket igényelhetik:

ADAT könyvtár

- Új könyvtár létrehozása (AU)

- Meglévő könyvtár törlése (AT)
- Meglévő könyvtár tartalmának módosítása (AM)
- Meglévő könyvtár módosítás nélküli kiolvasása (AO)

PROG könyvtár

- Új program beírása (PU)
- Meglévő program törlése (PT)
- Meglévő program módosítása (PM)
- Meglévő program kiolvasása és lefuttatása (PO)

Minden felhasználó rendelkezik egy számára meghatározott jogosultsággal, mely megadja, hogy az összes lehetséges műveletek közül melyeket veheti igénybe. Például:

0-s azonosítójú felhasználó	AU, AT, AM, AO és PO
1-s azonosítójú felhasználó	AO
2-s azonosítójú felhasználó	AO, PO
3-s azonosítójú felhasználó	AU, AT, AM, AO, PU,PT, PM, PO

Tehát a 0-s azonosítójú felhasználó az ADAT könyvtárat módosíthatja, a programkönyvtárat csak olvashatja. Az 1-es azonosítójú felhasználó mindkét könyvtárat olvashatja, de nem módosíthatja. A 3-as felhasználó viszont az összes művelet elvégzésére jogosult.

A felhasználók jogosultságait 1 bájt bitjeivel kódoljuk a következőképpen:

2^0	$\leftrightarrow$	AU
2^1	$\leftrightarrow$	AT
2^2	$\leftrightarrow$	AM
2^3	$\leftrightarrow$	AO
2^4	$\leftrightarrow$	PU
2^5	$\leftrightarrow$	PT
2^6	$\leftrightarrow$	PM
2^7	$\leftrightarrow$	PO

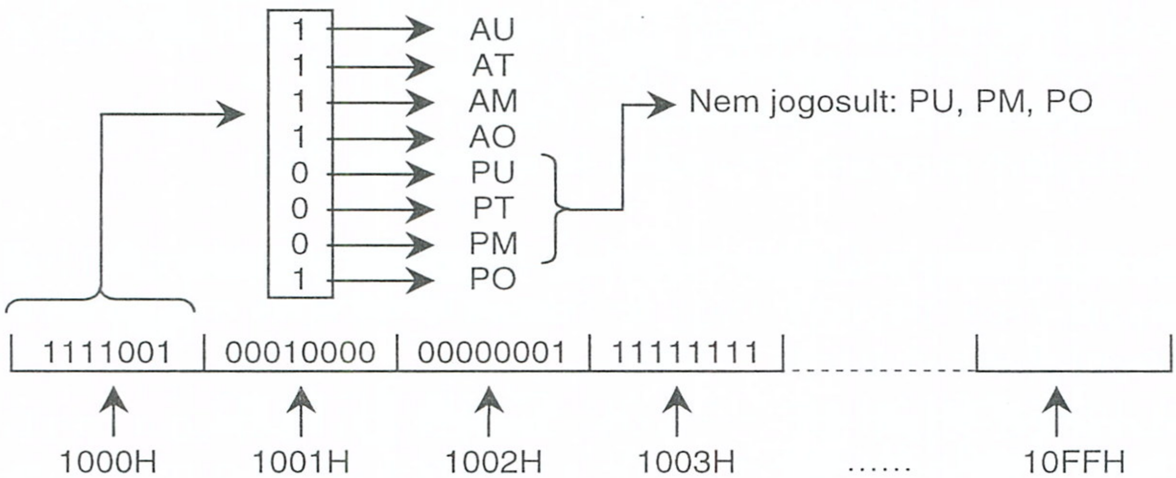
A bájt megfelelő bitje 1, ha az adott műveletre a felhasználó jogosult, 0 ha nem.

A felhasználók jogosultságait egy 256 bájtból álló tömbben helyezzük el a memória 1000H címétől kezdve folyamatosan. Tehát például:

1000H-s című bájt tartalmazza a 0-s azonosítójú felhasználó jogosultsági bájtját

1001H-s című bájt tartalmazza az 1-es azonosítójú felhasználó jogosultsági bájtját

stb.



VI.29. sz. ábra
A felhasználók jogosultságait tartalmazó bájtok elhelyezkedése a memóriában

Ha egy jogosultsági bájt tartalma bináris 0, akkor ez azt jelenti, hogy megfelelő sorszámú felhasználói azonosító és jogosultság nem került kiosztásra.

Írjunk programot a következő feladatra!

1. A program olvasson le két bájtot a billentyűzetről (0-s input port).
 - Az első bájt jelentse egy felhasználó azonosító sorszámát.
 - A második bájt pedig jelentse a felhasználó konkrét műveleti igényét, azonos módon bitenként kódolva, mint a jogosultsági bájtnál.

 (A különböző kódkonverziókat az egyszerűség kedvéért nem vesszük figyelembe.)

Fontos megjegyzések:

- Egy felhasználó nem mindig igényli az összes műveletet, melyre jogosult. Így az előzőekben rögzített jogosultságú 0-s sorszámú felhasználó kérheti például csak az AM

(adatkönyvtár módosítás) és az AO (adatkönyvtár megtekintése) műveleteket.
- Egy felhasználó tévedésből vagy szándékosan olyan műveletet is igényelhet, melyre nem jogosult.

2. Az FFh azonosítójú felhasználónak kitüntetett szerepe van (rendszergazda, supervisor). Feladata a jogosultsági táblázat módosítása. Ennek megfelelően, ha a beolvasott felhasználói azonosító FFh, akkor még egy bájtot be kell olvasni a 0-s input portról. Ez esetben a 2. és 3. beolvasott bájt értelmezése a következő:

1 bájt	FFh
2 bájt	felhasználói azonosító
3 bájt	jogosultság

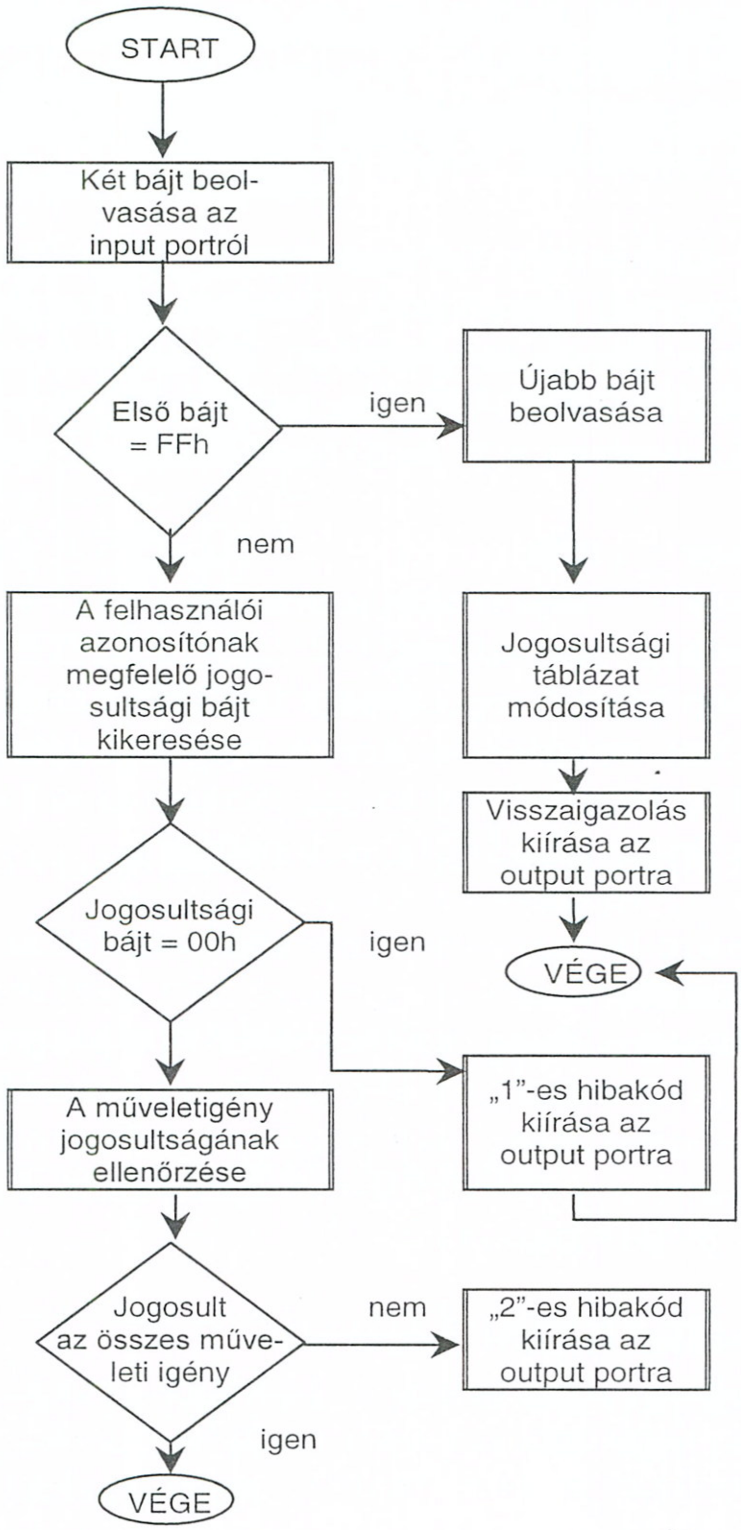
Ez esetben tehát a jogosultsági táblázatot módosítani kell a 2-es és a 3-as bájt alapján. Ennek megtörténtét követően a 2-es és 3-as bájt tartalmát visszaigazolásként ki kell írni a képernyőre. (Nem létező felhasználó azonosítója.)

3. A beolvasott felhasználói azonosítót ellenőrizni kell a jogosultsági táblázatban:
 - Ha a jogosultsági bájt tartalma 0 a program az 1 output porton keresztül írjon ki egy „1”-es hibakódot a képernyőre.
 - Ha a jogosultsági bájt alapján a felhasználó által igényelt legalább egy művelet nem megengedett, a program írjon ki egy „2”-es hibakódot.

A programozási feladat megoldása

A bonyolultabb programozási feladatok megoldásának egyik aranyszabálya, hogy azonnal nem kezdünk hozzá a program utasítássorainak leírásához (ezt nevezik programkódolásnak), hanem előbb
- megpróbáljuk nagyobb, viszonylag önálló részekre felbontani a programozási feladatot,
- átgondoljuk a feladat algoritmizálásának esetleges nehézségeit, súlyponti kérdéseit.

Feladatunk a VI.30. sz. ábra szerint bontható fel főbb funkcionális egységekre:



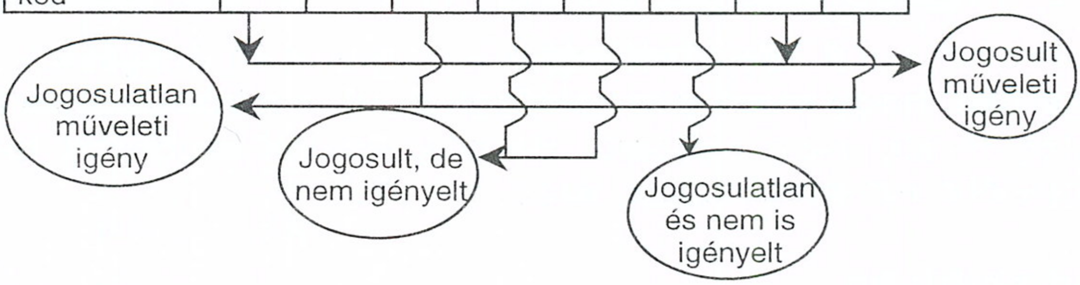
VI.30. sz. ábra
A programozási feladat algoritmusának funkcionális vázlata

Fő vonalakban tehát feladatunk nem tűnik túl bonyolultnak. Kezdjük most elemezni az algoritmust az ábra alapján! Ennek eredménye a következő lehet:

1. A blokksémában két helyen szerepel input művelet, három helyen output művelet. Minden I/O művelet előtt tesztelni kell a port állapotregiszterét, hogy az I/O műveletnek a feltételei adottak-e. Az ezekhez szükséges utasításokat nem szeretnénk többszörösen leírni. Döntés: az input és output műveletek végrehajtására szubrutint fogunk írni.
2. A funkcionális blokkséma alapján már el tudjuk képzelni a program lefutását, legfeljebb a „műveleti igény jogosultságának ellenőrzése blokk" adhat okot a fejtörésre. Ezt a részfeladatot alaposabban át kell gondolni.

Legyen a beolvasott műveleti igény bitenként kódolva és a jogosultsági bájt az ábra szerinti.

A bitek értelmezése	PO	PM	PT	PU	AO	AM	AT	AU
Műveleti igény	1	0	1	0	0	0	1	1
Jogosultsági kód	1	1	0	1	1	0	1	0



VI.31. sz. ábra
A műveleti igény és a jogosultsági kód bitenként kódolva

Az ábra alapján látható, hogy akkor kell a műveleti igényt jogosulatlannak minősíteni, ha legalább egy bitpozíción a műveleti igény 1 és ezen a pozíción a jogosultsági bit 0.

Hogyan lehet ezt eldönteni az utasításkészletben rendelkezésre álló utasításokkal?

Rövid vizsgálat alapján megállapíthatjuk, hogy ilyen utasításunk az utasításkészletben nincs.

Hívjuk most segítségül a Boole-algebrát. Jelölje „*A*" a műveleti igény bitjét, „*B*" pedig a jogosultsági kód bitjét. Jelöljük 1-gyel azt az esetet, ha a műveleti igény jogosult, 0-val pedig azt, ha nem. Ekkor a VI.32. sz. ábra szerinti igazságtáblázatot kapjuk.

A	1	1	0	0
B	0	1	1	0
Jogosult / Nem jogosult	0	1	1	1

VI.32. sz. ábra
A jogosultságellenőrzés mint logikai művelet

Keressük ki ezt a műveletet a II.2. sz. táblázatból (11 kimenet), és ez alapján a II.3. sz. táblázatból megállapíthatjuk, hogy a keresett művelet az $\overline{A}+B$, azaz a „nem" „*A*" vagy „*B*".

Ez a logikai művelet a Ha *A*, akkor *B* amit implikációnak nevezünk. Jelölése *A* → *B*.

Ha a műveleti igény bájtnegáltjára és a jogosultsági bájtra alkalmazva az $\overline{A}+B$ logikai műveletet, az eredmény FFh, akkor a műveleti igények között egyetlen jogosulatlan sincs. Az egyszerűbb feltételes elágazás miatt (BZ = ugrás, ha ZF = 1 azaz az eredmény 0) ehelyett az $\overline{\overline{A}+B}$ kifejezést fogjuk a programban vizsgálni.

Kérdés, hogy most már nekifoghatunk-e a kódolásnak?

Még mindig nem célszerű. Ha programunk szubrutint tartalmaz, akkor előbb ezt specifikáljuk, vagy írjuk meg.

Input szubrutin

Neve: SINP

Feladata: A szubrutin kiolvassa a 0-s port állapotregiszterét. Megállapítja, hogy a port pufferregiszterében van-e új adat. (A port állapotregiszterének specifikációját az 1.3. pont tartalmazza.) Ha nem, akkor folytatja a tesztelést, ha igen kiolvassa a pufferregisztert AL-be, majd visszaugrik. A szubrutin csak AX értékét változtatja meg.

```
SINP:   XOR   AX, AX    ; AX törlése
SISM:   IN    1         ; Az állapotregiszter tartalmának
                        ; átvitele AL-be
        CLC             ; carry flag törlése
        RCR   AX        ; AX forgatása jobbra
                        ; a 2^0 bit a CF-be.
        JC    SEGY      ; ugrás ha CF = 1
        JMP   SISM      ; ugrás állapotregiszter újabb
                        ; beolvasásához
```

```
SEGY:   CLC
        RCR   AX
        JNC   SISM          ; ugrás, ha CF = 0
        IN    0             ; 1 bájt átvitele AL-be
                            ; a 0-s port pufferjéből
        RET                 ; visszaugrás szubrutinból
```

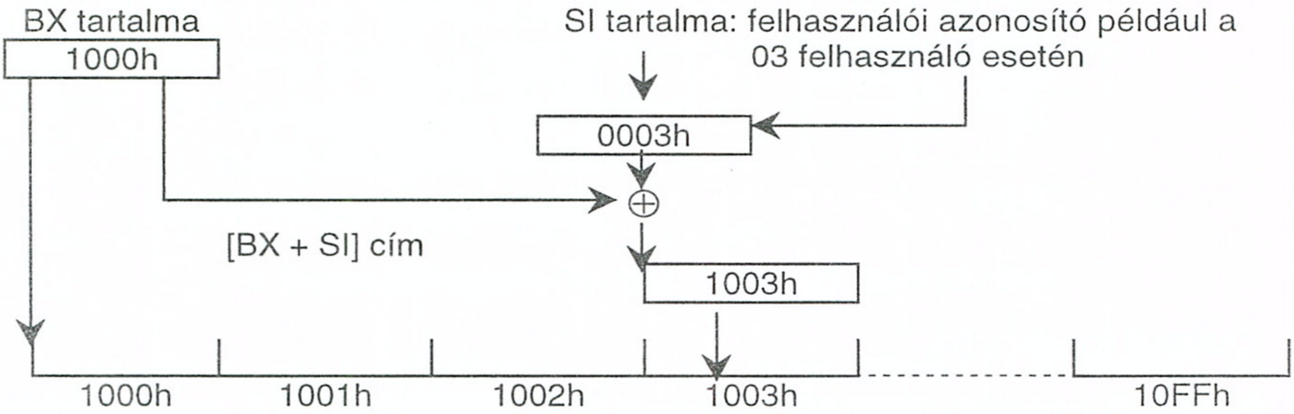
Output szubrutin

Teljesen hasonló megfontolásokkal építhető fel a SOUT nevű 1 bájtot kiíró szubrutin. A szubrutin paraméterként kapja AL-ben a kiírandó bájtot. (Feltételezi, hogy AX AL-től eltekintve bináris 0.)

```
SOUT:   PUSH  DX            ; DX kimentése verembe
        XOR   DX, DX        ; DX törlése
        MOV   DL, AL
SISM1:  IN    3             ; 1 port állapotregisztere
                            ; AL-be
        CLC
        RCR   AX
        JC    SEGY1
        JMP   SISM1
SEGY1:  CLC
        RCR   AX
        JNC   SISM1
        MOV   AL, DL
        OUT   2             ; 1 bájt kiírása
        POP   DX
        RET
```

A főprogram megírása

Most már hozzákezdhetünk a főprogram kódolásához, csak azt kell még eldöntenünk, hogyan címezzük a jogosultsági bájtokat tartalmazó tömböt. Bázisrelatív indexelt címzést fogunk alkalmazni, ami azt jelenti, hogy a BX regiszter fogja tartalmazni a tömb kezdőcímét, és az SI regiszter a felhasználói azonosítónak megfelelő indexértéket (lásd VI.33. sz. ábra).



VI.33. sz. ábra
A jogosultságokat tartalmazó tömb címzése

Ezt figyelembe vevő főprogramunk a következő:

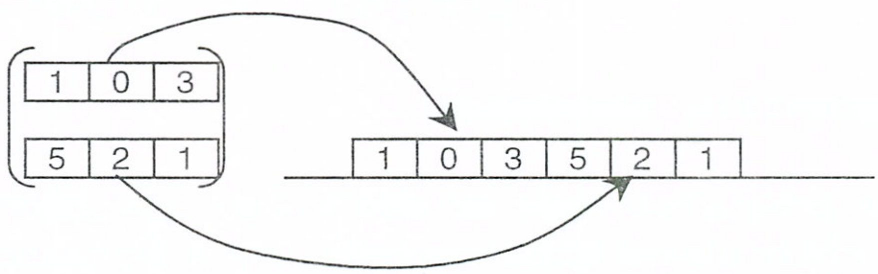
```
        MOV   BX, 100H       ; BX beállítása a kezdőcímre
        XOR   CX, CX
        XOR   AX, AX         ; CX, AX, DX törlése
        XOR   DX, DX
        CALL  SINP
        MOV   DL, AL
        CALL  SINP
        CMP   DL, FFH        ; Első bájt FFh?
        JE    MOD
        MOV   SI, DX
        MOV   CL, [BX+SI]
        CMP   CL, 00H        ; Jogosultsági bájt 0?
        JE    HIBA1
        NEG   AL             ; A Boole-algebrai
        OR    CL, AL         ; kifejezés kiszámítása
        NEG   CL             ; Nem jogosult
        JZ    VEGE
        MOV   AL, 02H        ; 2-es hibakód AL-be
        CALL  SOUT
        JMP   VEGE
HIBA1:  MOV   AL, 01H
        CALL  SOUT
        JMP   VEGE
MOD:    MOV   DL, AL         ; Felhasználói azonosító
        CALL  SINP           ; DL-be
        MOV   SI, DX
```

```
        MOV   [BX + SI], AL   ; Jogosultsági bájt módosítása
        MOV   CL, AL
        MOV   AL, DL
        CALL  SOUT            ; Felhasználói azonosító
        MOV   AL, CL          ; kiírása
        CALL  SOUT            ; Jogosultság kiírása
VEGE:
```

5.2. Tömbök kezelése a programokban

A programozásban nagyon gyakran szükség van sorokból, illetve oszlopokból álló számtáblázatok (ún. vektorok és mátrixok) kezelésére. Először vizsgáljuk meg, hogy ezeket a számtáblázatokat milyen módon tárolhatjuk le a memóriában. (A vektortömbök tárolását és indexelését az előző példában már bemutattuk.) !

Mivel a memóriában a számok csak sorfolytonosan helyezhetők el, a mátrixtömböket csak egymás utáni vektorokként tudjuk tárolni. Ennek elvét érzékelteti a VI. 34. sz. ábra.



VI.34. sz. ábra
Egy 2x3-as mátrix tárolása a memóriában

A programban a mátrixokat általában két regiszter segítségével címezzük, egyik regiszter tartalmazza a sorindexet, a másik pedig az oszlopindexet. !

Válasszuk például sorindexnek az SI, oszlopindexnek pedig a DI regisztereket.

Tekintsük az előző VI.34. sz. ábra példáját, és az egyes mátrixelemeket tároljuk kétbájtos számként. Tegyük fel továbbá, hogy a mátrix első sorának első eleme az 1000h című memóriaterületen helyezkedik el. Ekkor a számok memóriában való tárolását és címeiket a VI.35. sz. ábra mutatja.

00 01	00 00	00 03	00 05	00 02	00 01
1000h	1002h	1004h	1006h	1008h	100Ah

VI.35. sz. ábra
Egy 2x3-as mátrix tárolása a memóriában

Az SI-ben lévő sorindex 0, 1 (első és második sor) értékét veheti fel.

Az DI-ben lévő oszlopindex megengedett értékei 0, 1, 2.

Például a második sor első elemét (SI=1, DI=0) a következőképpen címezhetjük:

M (2,1) = A tömb kezdőcíme + SI • (3 • 2) + DI

Vegyük észre, hogy itt SI szorzótényezőjét úgy kapjuk meg, hogy az egy sorban elhelyezkedő elemek számát megszorozzuk az egy elem elhelyezéséhez szükséges tárolóterület bájtokban számított nagyságával.

Programozási mintafeladat

Írjunk programot, mely az előzőekben bemutatott pozitív elemű 2x3-as mátrix elemei közül meghatározza a legkisebb értékűt és azt elhelyezi az akkumulátorban.

A program megtervezésének alapgondolatai:

- A mátrixot tartalmazó tárolótömb kezdőcímét tartalmazza az SI regiszter. SI-t a következő lépésben a 2. sor kezdőcímére fogjuk beállítani.
- A DI regisztert a sorok egyes elemein léptetjük 2 bájtonként.
- Két ciklust szervezünk, a külső ciklus a sorok száma szerint, a belső ciklus az oszlopok száma szerint ismétlődik.
- Felveszünk egy munkaregisztert (DX), melybe betöltjük az első elemet. Ezt sorban összehasonlítjuk a mátrix elemeivel, és amennyiben egy elem kisebb a munkaregiszter tartalmánál, akkor ezt visszük át a munkaregiszterbe.

```
        MOV  SI, 1000H   ; A regiszterek kezdő
                         ; értékre állítása
        MOV  AX, 2       ;
        MOV  DX, [SI]    ; Első mátrixelem
UJRA:   MOV  CX, 3       ;
        MOV  DI, SI
HAS:    CMP  DX, [DI]
        JB   KIS
        MOV  DX, [DI]
KIS:    INC  DI
        INC  DI
        LOOP HAS
        ADD  SI, 6       ; SI beállítása
        DEC  AX          ; a következő sor
        BNZ  UJRA        ; kezdőcímére
```

Külső ciklus: UJRA … BNZ UJRA

belső ciklus: HAS … LOOP HAS

Ajánljuk az olvasónak, hogy az előző programrészlet lefuttatását" papíron nyomonkövetve a regisztertartalmak változtatását végezze el. (Ezt szokták nevezni „száraz" tesztnek.)

? Ellenőrző kérdések

1. 64 Kbájtos főtárhoz hány bites címeket kell használni?
2. 1 Mbájtos tároló esetén hány címvonalat tartalmaz a címsín?
3. Milyen szempontokat kell mérlegelni egy processzor regiszterkészletének megtervezésekor?
4. Milyen regisztereket tartalmaz a modellprocesszor?
5. Mit jelölünk AL, BL, CL és DL – el?
6. Milyen utasítások veszik figyelembe a FLAGS regiszter bitjeit?
7. Milyen flag-eket tartalmaz a FLAG regiszter?
8. Modellszámítógépünkben milyen regisztereket tartalmaznak az I/0 portok?
9. Hogyan azonosítjuk (címezzük) a regisztereket a gépi utasításokban?
10. Mit kell figyelembe venni egy processzor címzési módjainak megtervezésekor?
11. A memóriának milyen címzési módjai vannak modellszámítógépünkben?
12. Hogyan címezzük az I/0 portok regisztereit?
13. Hogyan különböztetjük meg a címzési módokat?
14. Milyen elvek alapján tervezhetjük meg a gépi utasítás szerkezetét?
15. Miért lehet kihagyni az utasításszerkezetből az eredményadat címét?
16. Miért lehet kihagyni az utasításszerkezetből a következő utasítás címét?
17. Miért kell törekednünk az utasításszerkezet kialakításakor a rövidebb utasítások definiálására?
18. Ismertesse a modellprocesszor utasításainak alapszerkezetét!
19. Milyen utasítástípusokat ismer a modellprocesszorunk?
20. Sorolja fel a modellprocesszor aritmetikai utasításait!
21. Sorolja fel a modellprocesszor logikai utasításait!
22. Sorolja fel a modellprocesszor adatmozgató utasításait!
23. Sorolja fel a modellprocesszor vezérlésátadó utasításait!
24. Hogyan tudja eldönteni a processzor a gépi kódú utasításról, hogy hány című?
25. Ismertesse milyen lépésekkel lehet meghatározni egy összeadó utasítás gépi kódját!
26. Milyen megfontolások alapján tudjuk a 6 számot összeadó programot rövidebben megírni?
27. Mit jelölnek egy Assembly programban a szimbolikus címek?

28. Mit nevezünk ciklusnak?
29. Milyen kódtáblázatokat használ fel az Assembler?
30. Hogyan fordítja le a fordítóprogram a szimbolikus címeket?
31. Mit nevezünk Assemblernek?
32. Hogyan történik az ADD utasítás végrehajtása?
33. Mi történik a SUB utasítás végrehajtása során?
34. Milyen célt szolgál a CMP utasítás?
35. Hogyan jelöljük az Assembly programnyelven a hexadecimális és a bináris számokat?
36. Ismertesse az INC és a DEC utasításokat!
37. Hogyan lehet egy szám abszolút értékét előállítani programban?
38. Hogyan működik az „AND" és az „OR" utasítás?
39. Ismertesse az „XOR" és a „NOT" utasításokat!
40. Mi az adatmozgatás Assembly utasítása és milyen utasítástípusok a megengedettek?
41. Mit tudunk megoldani a POP és a POPF utasításokkal?
42. Milyen utasításokkal lehet a regiszterek tartalmát verembe menteni és hogy történik ennek az utasításnak a végrehajtása?
43. Hogyan és hova mentjük ki a regisztertartalmakat a szubrutinokban?
44. Ismertesse a LOOP utasítás működését!
45. A 100 számot összeadó programban értelmezze az
 ADD AX, [SI] utasítást!
46. Hogyan hajtja végre a processzor a CALL utasítást?
47. Mi történik a RET utasítás végrehajtásakor?
48. Mi történik, ha a RET utasítás előtt közvetlenül egy PUSH AX utasítás áll? Lefuthat hibátlanul a program?
49. Ismertesse a JMP utasítás végrehajtását!
50. Milyen feltételeket tudunk tesztelni a feltételes ugróutasításokkal?
51. A

    ```
    CMP   AX, BX
    JNE   TOV
    ```

 utasítások esetén mikor folytatódik a program a TOV: címkéjű utasítással?
52. Milyen célra szolgálnak a kapcsolók? Mondjon egy példát a kapcsoló programban való felhasználására!
53. Mire lehet használni az INT és az IRET utasítást?
54. Mit jelent egy regiszter léptetése?
55. Mit jelent egy regiszter forgatása?
56. Mi a SHR és az SHL utasítás végrehajtásának hatása?
57. Mi az RCR és az RCL utasítás végrehajtásának hatása?

58. Ismertesse az RCR utasítás működését!
59. Milyen két csoportba sorolhatók a processzorvezérlő utasítások?
60. Milyen esetben folytatódik a program, ha a HLT utasítással leállítottuk?
61. Milyen könyvtárak vannak komplex programozási mintafeladatunkban és ezekkel milyen műveleteket végezhetnek a felhasználók?
62. Hogyan kódoljuk a felhasználók jogosultságát mintafeladatunkban?
63. Ismertesse a programozási mintafeladatot!
64. Hogyan fogunk hozzá egy programozási feladat megoldásához?
65. Mi indokolja mintafeladatunkban az I/O műveletek szubrutinnal való megírását?
66. Milyen logikai művelettel ellenőrizhetjük a felhasználók műveleti igényének jogosultságát?
67. Milyen programrészeket specifikálunk (kódolunk) először?
68. Hogyan címezzük a jogosultsági bájtokat tartalmazó tömböt a főprogramban?
69. Hogyan tároljuk a mátrixtömböket a memóriában? Ennek elvét mutassa be egy példán!
70. Hogyan címezzük a mátrixelemeket programozási mintafeladatunkban?
71. Mutassa be, hogy milyen algoritmussal lehet meghatározni egy mátrix legkisebb elemét!

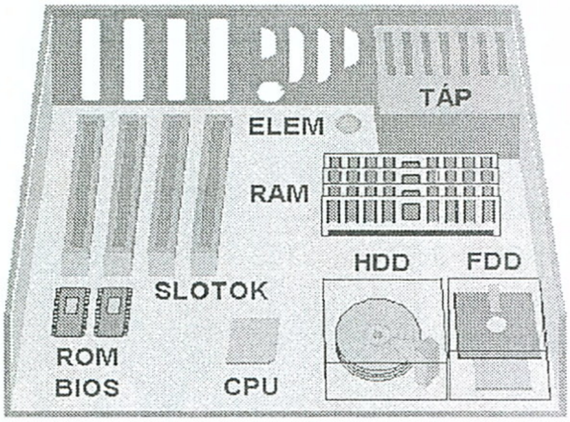
VII. fejezet

AZ IBM PC A GYAKORLATBAN

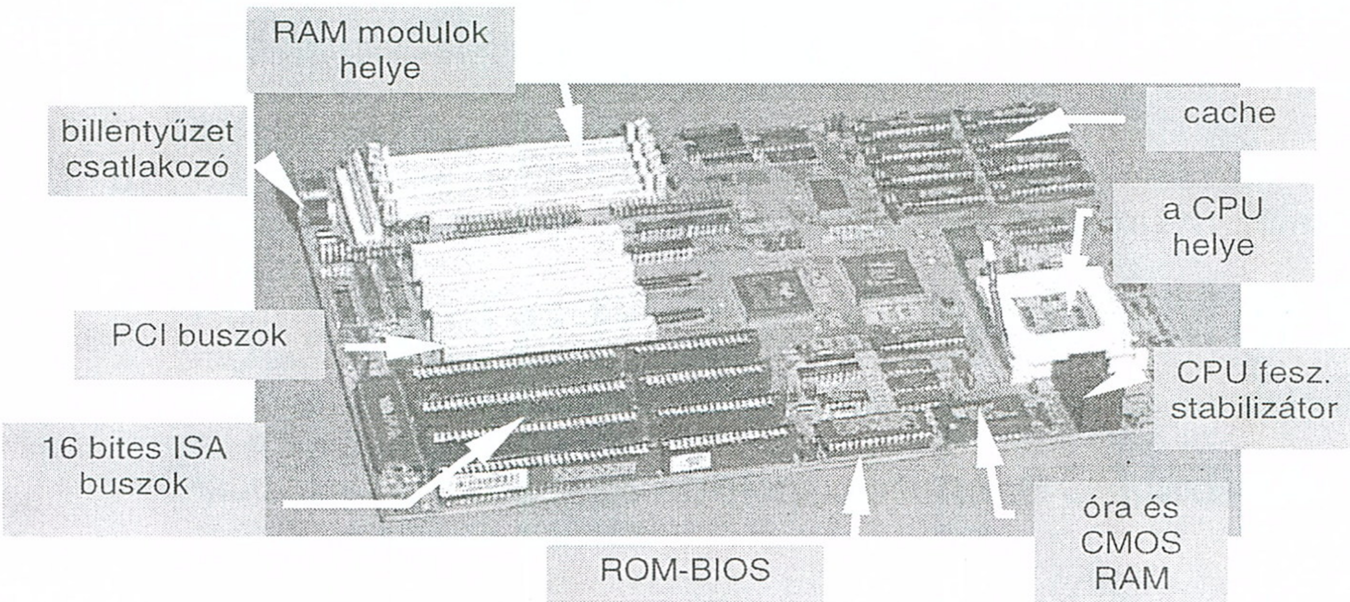
1. Az IBM PC számítógéptípus részei

Az 1981-ben megjelent **IBM PC** egy nagyon fontos tulajdonságában különbözött az abban az időben kapható többi személyi számítógéptől (*PC – Personal Computer, személyi számítógép*): hasonlóan a nagygépekhez, *moduláris* felépítésű volt, azaz belseje nem volt "gyantával kiöntve", több részegységből állt, így tetszőleges módon lehetett az egyes darabokat cserélni, bővíteni. (Általában, ha egy egység elromlik, nem is szokták megkeresni a hibát, hanem az egészet "kidobással javítják").

A *nyomtatott áramköri kártyás* egységek (*NYÁK*) közül a legnagyobb és legfontosabb az ún. **alaplap** (angolul *main board* vagy *mother board*). Ezen helyezkedik el a mikroprocesszor és minden mást (billentyűzet, memória, háttértárak stb.) az itt található különböző típusú aljzatokba kell csatlakoztatni.



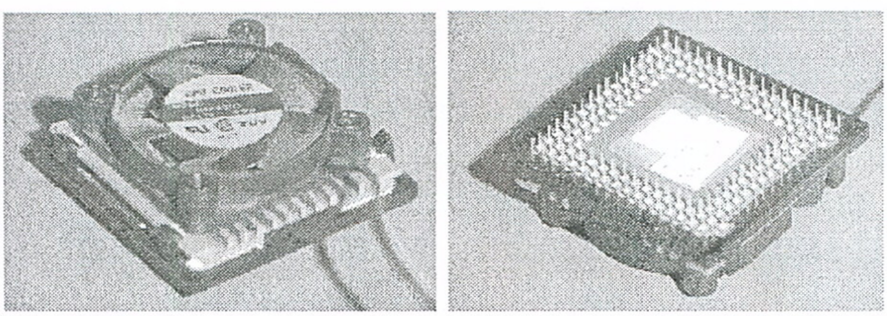
A számítógép belseje



Régebbi, Pentium CPU-val felszerelt alaplap

Eddigi ismereteink összefoglalásaként tekintsük át röviden az egyes egységek feladatait!

1.1. Mikroprocesszor (Central Processing Unit)

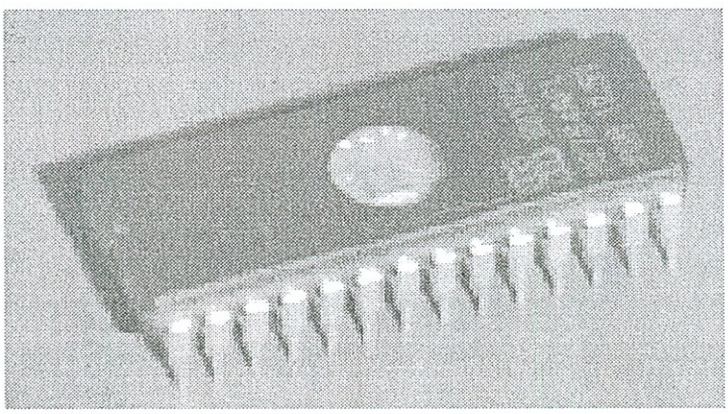


Régebbi típusú mikroprocesszor hűtőbordával és ventillátorral

A számítógép legfontosabb része a központi egység, nevének rövidítése ***CPU****. Ez a chip önállóan semmit nem tud csinálni. Feladata végrehajtani azokat a feladatokat, utasítássorozatokat, amelyeket* ***gépi kódú program*** *formájában írunk elő neki.* A számítógépet felhasználó emberrel sem tud kommunikálni – nem értenénk meg a jeleit és ő sem a miénket. Ezért számtalan kiegészítő berendezés veszi körül, amelyek ezt a kétirányú kommunikációt biztosítják. Ilyen pl. az adatbevitelt biztosító billentyűzet vagy egér és az eredményeket megjelenítő monitor vagy nyomtató. A CPU az alaplapon belül önállóan cserélhető egység, a jelenleg kapható típusokon a melegedés ellen hűtőbordát és/vagy egy kis ventillátort szerelnek rá. Fontos jellemzője az órajel, mely a gyorsaságára utal. Minél nagyobb ez a szám, annál gyorsabb. A legelső IBM PC-k CPU-ja 4,77 MHz órajelfrekvencián dolgozott (kb. 4 millió elemi utasítás másodpercenként), a későbbi IBM AT-k 6 MHz-en, a jelenlegiek 266 MHz, illetve még magasabb frekvencián.

Mint már tudjuk a CPU belsejében találhatók az ún. *regiszterek*. Ezek egy-egy szám tárolására szolgáló memóriarekeszek, a gépi kódú programok használják fel adattárolásra, műveletvégzésre, vagy memóriacímzésre. Az 1997. januárban bejelentett Intel Pentium MMX processzor 4.5 millió tranzisztorból épül fel.

1.2. ROM



Erasable Programmable Read Only Memory (EPROM)
Elektronikusan törölhető és újraprogramozható memória.

A ROM a *Csak Olvasható Memória* (*Read Only Memory*) szavak rövidítése, azaz a benne eltárolt információt ki tudjuk nyerni, megváltoztatni viszont – speciális eseteket kivéve – nem lehet. Itt felmerülhet a kérdés: miért van szükség egy megváltoztathatatlan tartalmú adattároló eszközre, amikor a felhasználó különböző feladatokra

használja a számítógépet, mindig másra? *A ROM tartalma nem módosítható, így a számítógép bekapcsolásakor is rendelkezésre áll! Ezt használjuk ki, a bekapcsoláskor teljesen "üres" számítógép innen veszi az indításhoz szükséges információkat.*

Azaz a ROM a következőket tartalmazza:

- egy gépi kódú programot, amelyet a számítógép a bekapcsoláskor hajt végre. E rövid program egy öntesztet végez (*Power On Self Test – POST*), majd operációs rendszert keres a háttértáron – merevlemezen, esetleg floppyn vagy CD-n. Ha megtalálta, átadja a vezérlést. Ha nem talált semmit, hibaüzenetet ír ki a monitorra és a gép leáll, nem használható tovább.
- több tíz – szintén gépi kódú – programot (ún. szubrutint), amelyet a felhasználói programok meghívhatnak, felhasználhatnak. E programok biztosítják a legalapvetőbb kommunikációt a hardver egységekkel, pl. beolvasnak a billentyűzetről egy karaktert, beállítják a monitor különböző üzemmódjait, adatokat olvasnak be a háttértárról illetve kiírnak rá stb. Szokás ezért az alaplapi ROM-ot *ROM-BIOS*-nak is nevezni (*Basic Input Output System*).
- a monitoron megjelenő karakterek alakleírását, ezt azonban a monitorvezérlő kártyán található ROM tartalmazza.

A ROM mérete néhány tíz kilobájt. A benne található gépi kódú programok gyártótól függően más és más felépítésűek lehetnek, a külső belépési (kapcsolódási) pontokat viszont szigorúan betartják. Pl. a számítógép bekapcsolásakor a ROM egy meghatározott címétől (FFFF0 hexa, azaz az 1.048.560-as decimális rekesztől) kezdődő utasításokat hajtja végre a CPU.

A ROM tartalma bájtonként érhető el. Ez annyit jelent, hogy pl. olvasáskor a CPU az ún. címsínen keresztül kiadja a kívánt rekesz sorszámát, majd egészen addig vár, amíg az adatsínen megjelenik az adott sorszámú rekeszben található bájt tartalma. A kívánt rekesz számának kiadásától a tartalom megjelenéséig eltelt időt *elérési idő*nek nevezzük. A ROM memóriák elérési ideje nagyságrendileg 200 nanoszekundum (minél kisebb a szám, annál gyorsabb a memória).

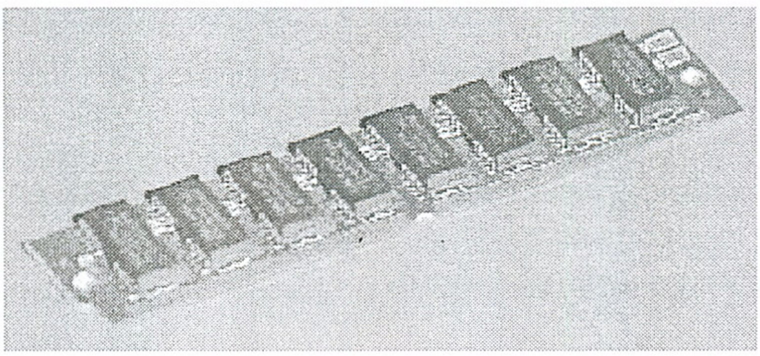
Mint tudjuk a számítástechnika a kettes számrendszert használja: *1 bit (binary digit) 0 vagy 1.* Előnye, hogy elektronikusan könnyű megvalósítani. Hátránya viszont a kis információsűrűség. Ezért a biteket nyolcasával csoportosítják: *1 bájt (byte) 8 bit*, két szélsőértéke 00000000 és 11111111. Ez 256 lehetőség, *1 bájt emiatt egy 0 és 255 közé eső egész szám, a határértékeket is beleértve*. Nagyobb egysé-

gei a *kilobájt*: 1 KB = 1024 bájt, *megabájt*: 1 MB = 1024 KB, *gigabájt*: 1 GB = 1024 MB; de *kilobit:* 1 Kb = 1024 bit. Az 1024 kettő egész számú hatványa, így a kettes számrendszerben használata "természetes", a 10 (negatív) hatványait tudjuk nehezen ábrázolni.

A ROM-BIOS az alaplapon található, ezen kívül más kártyák is tartalmazhatnak ROM memóriát (pl. hálózati-, monitorvezérlő kártyák stb.). Egyes hordozható számítógépeken az operációs rendszer is ROM-ban található – így biztonságos, mert egyrészt letörölhetetlen, másrészt a vírusok sem tudják megváltoztatni.

Az EPROM-ok helyett mostanában a más technológiával készült FLASH memóriákat használják, amelyek elektromos táplálás nélkül is megőrzik az információt és elektronikusan újraírhatók. Így pl. menet közben is frissíthető a FLASH ROM-BIOS a felhasználó által, míg EPROM alkalmazása esetén ez nem megvalósítható.

1.3. RAM



Régi, paritásvédelem nélküli 4 MB-os SIMM RAM

A **RAM** a *Véletlen Elérésű Memória* (*Random Access Memory*) szavak rövidítése. Ez nem azt jelenti, hogy a tartalmához egyszer hozzá tudunk jutni, egyszer pedig nem, inkább azt, hogy bármelyik rekesz tartalma *azonos idő alatt* kérdezhető le, illetve változtatható meg (helyesebb fordítása inkább *Közvetlen vagy Tetszőleges Elérésű Memória*). A RAM írható és olvasható, azaz tartalmát a CPU le tudja kérdezni és meg tudja változtatni.

A RAM az ún. **operatív tár**, vagy *munkatár*. Itt futnak a programjaink. Azaz egy bármilyen felhasználói program elindításakor a programot a számítógép betölti a RAM-ba és innen futtatja. Egy újabb feladatkor pedig a másik program kerül be a RAM-ba, azaz mindig más és más, az igényeknek megfelelő. A már említett POST többek között egy RAM tesztet is tartalmaz – hiszen nagyon fontos e memó-

riatípus hibátlan működése. A legtöbb gépen a RAM memóriát páratlan paritásvédelemmel védik, hiba esetén a rendszer leállítja a gép működését.

A RAM memória tartalma szintén bájtonként érhető el. Két típusa ismeretes, a **statikus** és a **dinamikus** RAM. A statikus RAM egy bitjét több áramköri elemmel tudjuk eltárolni (ún. *R-S tároló*, vagy *flip-flop* -pal), ezért drágább. A dinamikus RAM kevesebb alkatrészből áll (1 bitet 1 tranzisztor és egy kondenzátor jellegű elem tárol), felépítése miatt viszont állandóan frissíteni kell, amely járulékos áramkörök kiépítését teszi szükségessé. SRAM elemekből épülnek fel a CPU regiszterei, a gyorsítótár (cache) és speciális memóriák, pl. a PCMCIA memóriakártya. DRAM memóriából épül fel a PC-ken használt operatív tár (amit egyszerűen csak "RAM"-nak nevezünk).

A RAM-ot az alaplapon található aljzatokba kell csatlakoztatni. Nagysága a mostani számítógépeken általában 16-32 MB. Elérési ideje kisebb a ROM-nál (SRAM 15-20 ns, DRAM 60-70 ns). Jelenleg használt változatai a SIMM (Single In-line Memory Module), vagy a DIMM (Dual In-line Memory Module). A memóriaműveletek mindig is lassították a CPU-k működését, mert az igényelt rekesz tartalmának megérkezéséig áll a program végrehajtása. Az újfajta EDO-RAM (Extended Data Output DRAM) olyan újfajta dinamikus memória, amellyel az adatkiolvasási idő lecsökken, az SDRAM (Synchronous DRAM) pedig a processzorsín és a memória sebességének szinkronizálásával növeli a teljesítményt.

1.4. Gyorsítótár (cache)

A RAM-ba történő írás-olvasás az SRAM egységekből álló, néhány 100 KB-os, gyors – 15 ns elérési idejű – gyorsítótáron (cache) keresztül történik. Így a RAM elérése bizonyos esetekben gyorsabb, pl. ha ugyanazokat a fájlokat többször olvassuk, az olvasás első alkalommal a háttértárról, utána viszont a cache-ből történik. Íráskor a mentés a cache-be kerül és a számítógép kevésbé leterhelt időszakában történik meg a tényleges mentés. A cache az alaplapon, illetve a CPU-ban is megtalálható.

1.5. Merevlemez (hard disk, fixed disk, winchester)

A RAM fontos tulajdonsága, hogy tartalmát kikapcsoláskor elveszti. Tartalma amúgy is mindig változik – éppen amilyen programot futtatunk. Ezért a többször felhasználni kívánt programjainkat és

munkánkat egy olyan helyre kell eltárolni – "menteni", ahonnan napok vagy évek múlva is elő tudjuk bányászni. Ez az ún. **háttértár**. A merevlemez egy nagykapacitású háttértár, amely a számítógép dobozában található, az alaplaphoz egy kábellel csatlakozik. A nagy kapacitás mint előny mellett hátrány, hogy mind a meghajtó, vagy **Hard Disk Drive – HDD**, mind a benne található lemez általában rögzített, nem cserélhető.

A háttértáron a tárolás egysége a **fájl** (*file*), magyarul állomány. Egy egész fájlt tudunk betölteni a memóriába, a teljes fájlt tudjuk másolni, áthelyezni vagy éppen törölni. (Természetesen a programok egy fájl tartalmát bájtonként is tudják kezelni.)

A winchesteren helyezkedik el az operációs rendszer (azaz számítógépünkre olyat tehetünk fel, amilyen jólesik, ha kedvünk van Windows 95-öt, 98-at, NT-t, LINUX-ot, OS/2-t vagy bármi mást).

Megjegyzés: bizonyos szempontból a merevlemezt is munkatárnak kell tekinteni. Előfordulhat, hogy egy-egy programot rosszul írtak meg és néhány fájlba beletöröl. Esetleg *véletlenül* egy hibás utasítás kiadásával mi törlünk. Vagy kapunk egy vírust. Vagy este kikapcsoljuk a számítógépet, reggel gyanútlanul bekapcsoljuk és nem történik semmi, a gép nem indul el, a winchesterhez nem tudunk hozzáférni! Egyszóval a merevlemez tartalma "bármikor" megváltozhat! Ezért javasolt, hogy mindent ami fontos, hajlékonylemezen is tároljunk, beleértve programokat és a nap végén elmentett munkánkat. Így ha a géppel bármi történik, egy másik gépbe behelyezve a hajlékonylemezt, munkánk nem vész el.

A legelső IBM PC-k még nem tartalmaztak merevlemezt, ezt az 1982-ben bemutatott XT (eXtended Technology)-ben használták először, mérete 10 MB volt. Kapacitása a jelenleg használt gépeken 500 MB – 9 GB közötti. Élettartama években mérhető (50 millió írási-olvasási ciklus). A merevlemez elérési ideje nagyságrendekkel nagyobb a ROM és a RAM memóriáknál (10-20 ezredmásodperc).

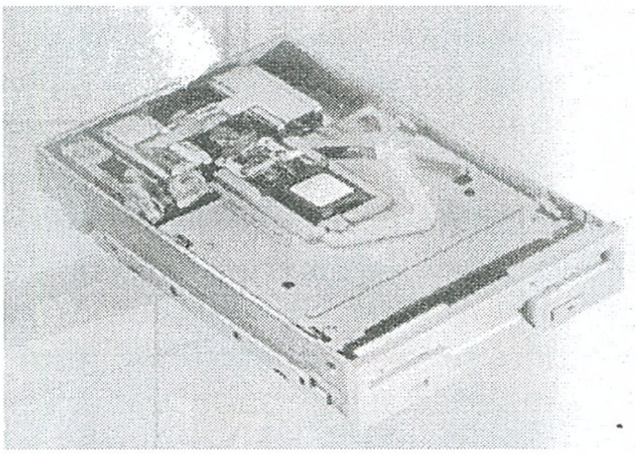
1.6. Hajlékonylemez (floppy disk)

A hajlékonylemez tehát a "legigazibb" háttértár, amely adataink tárolására szolgál. A meghajtóban (**Floppy Disk Drive – FDD**) cserélhetjük a lemezt. Hátránya a kis kapacitás, a jelenleg általánosan használt 3.5"-es lemezeknél 1.44 MB. Az adatok itt is fájlokban helyezkednek el. Élettartama szintén években mérhető (5.000 írási-olvasási ciklus), de a külvilág mágneses sugárzása és a kopás ha-

marabb tönkreteheti a benne tárolt információkat. Elérési ideje nagyobb, az adatátviteli sebessége (kb 50 KB/sec) jóval kisebb a merevlemeznél.

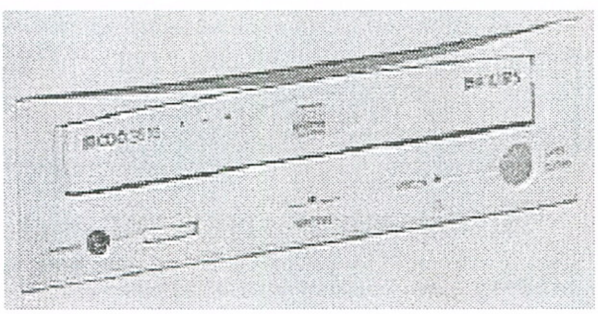
A későbbi bosszankodás elkerülése érdekeben érdemes az ún. *névtelen* (*noname*) lemezek helyett kicsivel több pénzt áldozni és márkásabb floppyt beszerezni – megéri!

Az utóbbi években kezd elterjedni a magneto-optikai elven működő ún. *A: DRIVE*, melyhez 120 MB kapacitású cserélhető lemezek tartoznak, ugyanakkor képes kezelni a hagyományos 1.44 MB-os floppykat is. Eleinte ígéretesnek tűnt, de – talán a DVD megjelenése miatt – egyelőre nem terjedt el annyira.

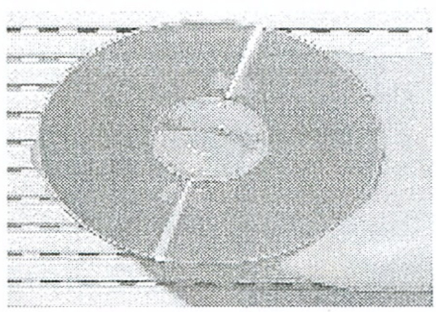


A régi floppymeghajtók lehetséges utódja, a felülről kompatibilis A:DRIVE

1.7. Optikai lemezek



CD író



CD lemez
Előállítási ára 1 USD alatt van

1982-ben a Philips és a Sony által kifejlesztetett **Compact Disc** három fő tulajdonságában tért el a hagyományos bakelit lemezektől:

- az adatok (zene) tárolása analóg helyett digitális módon történik, egy korong felületén láthatatlan, infravörös lézersugárral létrehozott digitális jelsorozattal,
- az olvasófej (lézer) nem ér hozzá a lemezhez,
- az olvasás nem kívülről befelé, hanem belülről kifelé történik.

Azaz a CD-n tárolt zene 0-k és 1-esek sorozata (pontosabban az adatátvitel másodpercenként 150 KB, mely mindkét hangcsatorna anyagát tartalmazza).

Az 1985-től gyártott CD-ROM már nem zenét tartalmaz, hanem számítógépes programot, adatokat. Az egyszeres meghajtók adatátviteli sebessége azonos a zenei CD-vel (150 KB/sec), a manapság használt 24x-es, 32x-es stb. meghajtók adatátviteli sebessége nagyjából 150-nek adott számszorosa. A CD-ROM egység természetesen képes lejátszani a zenei CD-t is (Compact Disc Digital Audio, CD-DA).

A 12 cm átmérőjű CD lemez kapacitása 650 MB (vagy 74 perc zene, a kettő ugyanaz: a 150 KB/sec-ból kiszámolható), az átviteli sebesség pedig kb. 2,5-5 Mbit/sec (*1 Baud nagyjából 1 bit/sec*). Hátránya és egyben előnye, hogy az adatok felülírása nem olyan egyszerű, mint a merev-, vagy a hajlékonylemeznél (Write Once Read Many – WORM lemezek: csak egyszer írható, a felvitt információ többször olvasható).

A CD élettartamát a mágneses adatrögzítési elvet használó merev- és hajlékonylemezek sokszorosára becsülik. Kisebb karcolásokat probléma nélkül elvisel. Elérési ideje viszont nagyobb a merevlemeznél (70-90 ezredmásodperc). A sok típus miatt általában nem olyan egyszerű a rendszerhez kötni, mind pl. a merevlemez-, vagy floppy meghajtót: a csatlakoztatáshoz szoftveres illesztőprogram telepítése is szükséges.

A CD technológia fejlesztés nem állt le. A Philips és Sony által kifejlesztett **HDCD** (High Density Compact Disk) dupla adathordozó réteg esetén 7.4 GB tárolására képes. A hangrögzítésben a magnókazetta (MC) és a "hagyományos" CD mellett elterjedőben van a szintén digitális adatrögzítő, a Sony MiniDisc (MD).

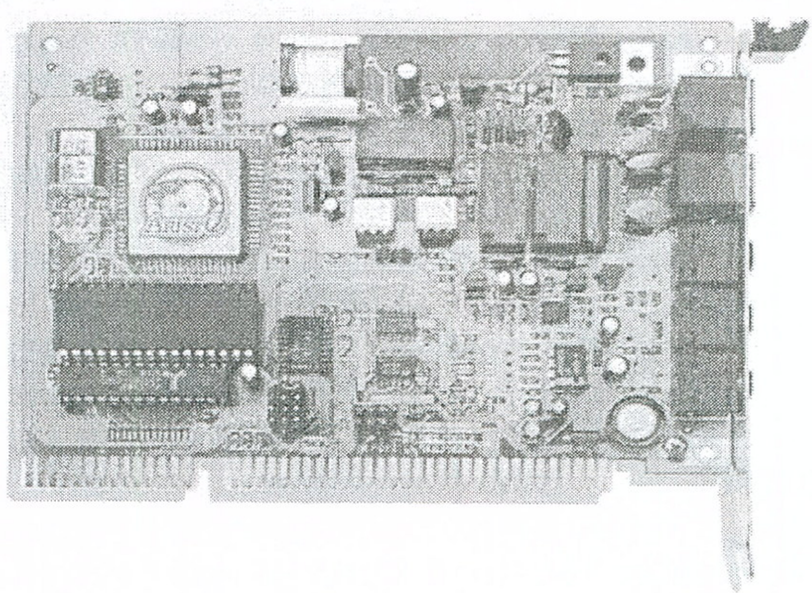
1995-ös fejlesztés a **DVD** (Digital Versatile Disk, digitális sokoldalú lemez), melynek kapacitása még nagyobb: az egyoldalas egyrétegűé 4.7 GB, a kétoldalas kétrétegű ROM lemezé pedig 17.8 GB. Mérete megegyezik a hagyományos CD lemezzel: 8 illetve 12 cm átmérőjű. A nagy kapacitás miatt kiválóan alkalmas mozifilmek kiváló minőségű tárolására, a meghajtók természetesen képesek a "régi" CD-k lejátszására is. A CD-ket a számítástechnikán kívül használják még filmvetítésre, Karaoke-ra és játszásra is (Sony Playstation).

1.8. Hangkártya

A számítógépben van egy beépített kis hangszóró (*PC-Speaker*), amelyet egyes régi játékprogramok használnak. A 7-es ASCII kódú karakter (Bell) röviden megszólaltatja a hangszórót. Az újabb programok azonban az alaplapra rácsatlakoztatható hangkártyát használják. Segítségével lehetőség nyílik hanganyag felvételére és CD minőségű sztereo lejátszására. Azonban túl sokat ne várjunk: minősége össze sem hasonlítható az elektroncsöves erősítő és elektrosztatikus hangfal által kiadott hanggal!

1.9. Modem

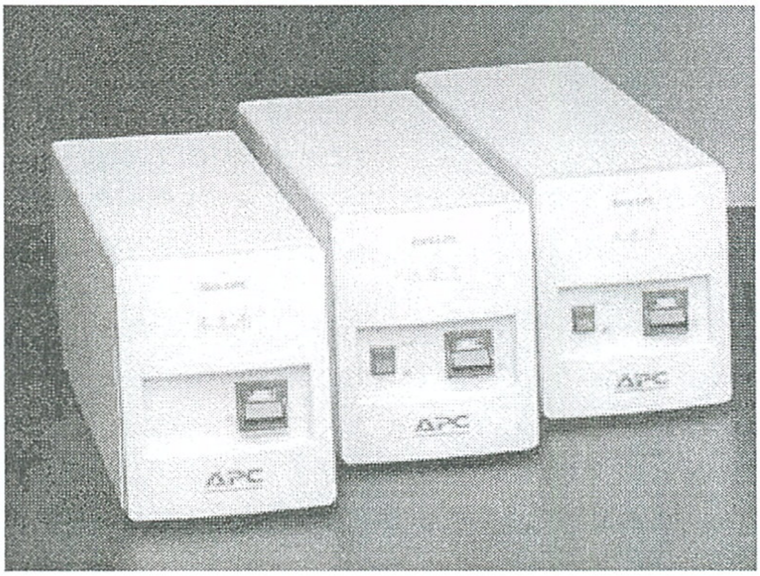
A *MOdulátor-DEMmodulátor* rövidítése. Otthoni használat esetén az Internetre számítógépünk telefonvonalon keresztül csatlakoztatható. A modem a telefonvonal analóg jelét alakítja át a gép számára értelmezhető digitális jellé és fordítva. Természetesen a telefonvonal túloldalán is található egy másik modem, amely az Internet szolgáltató számítógépéhez kapcsolódik. A modem egyik fontos adata az adatátviteli sebessége (pl. 56 kbit/sec). Létezik a számítógép házába szerelhető és kívülről rácsatlakoztatható változata.



Belső faxmodem kártya. A kártya alját az alaplapba csatlakoztatják, jobb oldalán jól láthatók a telefonkábel aljzatok

1.10. Tápegység (Power Supply), szünetmentes tápegység (Uninterrupted Power Supply, UPS)

A számítógép egységei törpefeszültségen működnek (2.8 vagy 3.3 vagy +/- 5V-on a digitális, +/- 12V-on az analóg áramkörök). A kb. 200 W-os kapcsolóüzemű tápegység a hálózati feszültséget alakítja át, majd egyenirányítják, szűrik és stabilizálják. Az így kapott feszültség egyenletes és szinte zavarmentesnek mondható. Megbízható működésű, leggyakrabban egyetlen mozgó alkatrésze, a hűtőventilátor szokott meghibásodni. Főként szerver gépeknél használják az UPS-t, mely egy külső egység és a hálózati feszültség kimaradását képes – általában néhány percig – áthidalni.



Szünetmentes tápegység (UPS)

1.11. CMOS RAM és akkumulátor

Ez a 64 vagy 128 bájtos RAM memória szintén az alaplapon helyezkedik el és tárolja a számítógép hardver felépítésének legfontosabb adatait (milyen háttértárak vannak a gépben stb.). Emellett tartalmazza a számítógép ún. **belső óráját**, amely a gép kikapcsolása alatt is működik. A **CMOS RAM** és a belső óra energiaellátását a régi alaplapokon egy kis akkumulátor látta el (amelyből évek múltán kifolyt a belső folyadék és tönkretette az alaplapot), az újabb gépekben egy bármely órásboltban kapható gombelem szolgál erre a célra.

A belső óra nem a legpontosabb. Tartalmát az operációs rendszerek és a felhasználói programok is felhasználják, ezért kb. havonta egyszer érdemes utánaállítani.

1.12. Egyéb kapcsolók

A számítógép dobozán a következőket találhatjuk:

egy *Power* feliratú kapcsolót, amellyel a gépet be- és kikapcsolhatjuk. A számítógép áramkörei nem szeretik, ha másodpercenként tízszer ki-be kapcsoljuk a gépet, ha egyszer ki kell kapcsolni, várjunk néhány másodpercet és csak utána kapcsoljuk vissza. Az ún. ATX-es alaplappal felszerelt gépek az operációs rendszerből (pl. Windows 95/98) történő kilépés után maguktól kikapcsolnak. *Bekapcsolás sorrend:* először kapcsoljuk be a perifériákat (nyomtató, scanner, monitor stb.) és a gépet kapcsoljuk be *legutoljára*! Kikapcsoláskor fordítva: először a számítógépet áramtalanítsuk!

egy *Reset* feliratú gombot. Ezzel végezhetjük a számítógép ***hidegindítás***át. Hatása majdnem olyan, mintha a gépet egy pillanatra ki-, majd bekapcsolnánk. Néhány típusról elhagyták.

egy *Turbo* feliratú gombot. A régebbi alaplapok két sebességgel tudták meghajtani a CPU-t, a gomb benyomásával állítottuk be a gyorsabbat. A jelenleg használt alaplapokon általában dísz.

LED kijelzőt, amelyen egy szám, pl. "`199`", esetleg szöveg, pl. "`HI`", "`LO`" szerepel. A szám a gép sebességére utal ("hány MHz-es"). Jó ha tudjuk, hogy csak tájékoztató adat, kis kapcsolókkal akármilyen kijelzést be lehet állítani.

2. A személyi számítógépekről

2.1. Rövid történet

A 70-es években jelentek meg az első személyi számítógépek, melyekből még több alapvető egység hiányzott. Ilyen volt pl. az Altair is. 1977-ben jelent meg az első *sorozatban gyártott* személyi számítógép, az Apple II. A 80-as évek elején a számítástechnikában a mai gépkocsiparkhoz hasonló kavalkád uralkodott: mindegyik vállalat megalkotta *saját* számítógépét, amelyek – bár hasonló alkatrészekből épültek fel, pl. a CPU néhány népszerű típus egyike volt – alig

tudtak egymással kommunikálni. Az IBM PC-k elterjedésüket annak köszönhették, hogy más gyártók "lehetőséget kaptak" ugyanolyan tudású *utánzatok*, ún. *klón*-ok gyártására. Elsősorban a Távol-Keleten nagyarányú dömping indult el, ez okozta azt, hogy *mára a személyi számítógépek piaca homogenizálódott, kb. 80%-ban IBM PC kompatíbilis gépekből áll.*

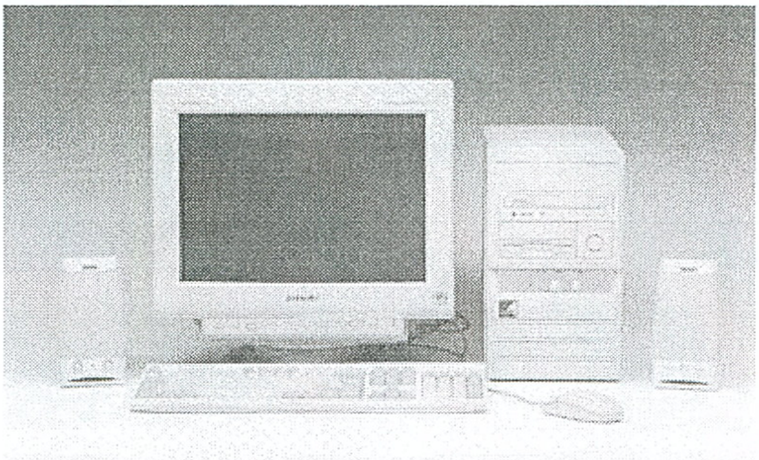




A 80-as évek két sikerszámítógépe: az angol Sinclair ZX Spectrum (1982) és a Commodore 64 (1982), a maguk idejében drága és nagy tudású gépnek számítottak. A Spectrum jellemzői: egy egységet alkot a billentyűzet és az „alaplap": 3.5 MHz-es Z80A CPU, 16 KV ROM, benne BASIC interpreter, 16 vagy 48 KB (!) RAM, kazettás magnetofon háttértár, egyszerű televíziós megjelenítés 256x192 képpont felbontás és 8 szín+két fényerősség+villogás, egybites hangszóró. Számtalan „nosztalgia"-weblap található e gépekről (pl. www.datanet.hu/sinclair, www.nvg.ntnu.no/sinclair, www.zock.com/8-Bit/D_Main.HTML).

A ***kompatibilitás*** *annyit jelent, hogy bár a számítógépek hardver felépítése (hardware – kemény áru) azaz az áramköri kialakítás gyártótól függően más és más, sőt pl. a ROM-ban található programok, azaz a szoftver (software – lágy áru) is eltérő, programozás szempontjából egyformának tekinthetők. Azaz egy IBM PC-re írt programot a világ bármely részén kifejleszthetik, az egész világon használható a PC kompatíbilis gépeken. Ez nagyfokú rugalmasságot és mobilitást jelent.*

2.2. Ma modern konfiguráció



Mai IBM PC kompatibilis számítógép konfiguráció

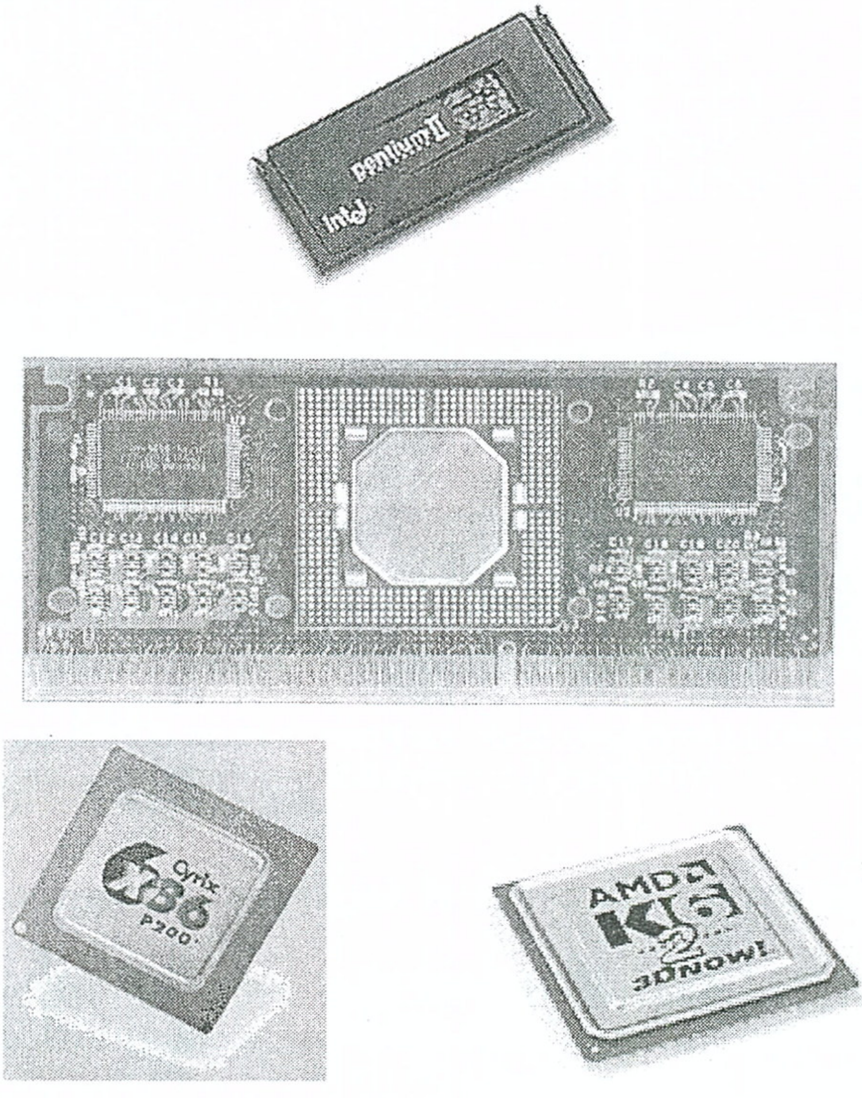
A számítógép LEGO rendszer: a felhasználási célnak és pénzünknek megfelelően több darabból állíthatjuk össze, az egyes egységeket később modernebbre cserélhetjük. Az összeállítást egyes üzletek elvégzik helyettünk.

A HARDVER

A jegyzet készítésének időpontjában (1999. nyár) modernnek számító konfiguráció az alábbi:

- Intel Pentium III vagy Pentium II CPU, legalább 266 MHz-es órajellel, 512 KB cache; természetesen a CPU származhat más gyártótól is, pl. AMD vagy Cyrix; ATX-es ház
- 32 vagy 64 MB SDRAM, EDO RAM vagy még újabb fejlesztésű RAM modul

- SVGA vagy XGA monitorvezérlő kártya (pl. 1024x768x16,7 millió szín, 4 vagy 8 MB video RAM), esetleg grafikus gyorsítókártya; valamint 14, 15 vagy 17"-os LR-NI (Low Radiation-Non Interlace) monitor
- min. 3 GB HDD
- 1.44 MB FDD
- 32x-es CD-ROM, esetleg CD író
- minőségi hangkártya (pl. Sound Blaster kompatíbilis)
- legalább 28.8 kilobit/sec modem az Internet eléréséhez (hálózat esetén helyette hálózati kártya)
- 102 gombos magyar billentyűzet
- 2 soros, 1 párhuzamos, 1 game port (esetleg PS/2, AGP, USB és SCSI csatlakozók)
- egér és egérpad

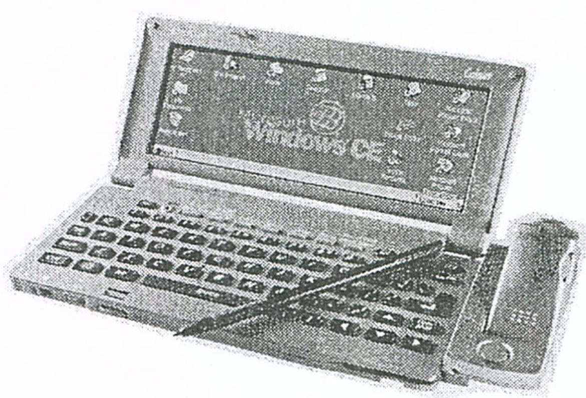


Modern mikroprocesszorok: Intel Pentium II kívülről és belülről, Cyrix 6x86, valamint AMD K6-2 3DNow!

A fenti konfiguráció újonnan és boltban legkevesebb 150 ezer Ft-ba kerül, felső határ a csillagos ég.

Egyre inkább terjednek a hordozható számítógépek, nevük csökkenő méretük szerint laptop-notebook-palmtop. Drágábbak asztali társaiknál, de tudásuk (mára) megegyezik. A hordozhatóság nagyfokú mobilitást és széles felhasználási kört jelent. Pl. jelzőlámpás kereszteződésekben a forgalomirányító berendezés újraprogramozását is végezhetik laptoppal. Ilyen előnyök mellett megengedhető egy fokozattal rosszabb konfiguráció is, pl. 486-os CPU-val felszerelt notebook még jól használható.

Egyébként a régebbi típusú CPU-kat nem szabad "leírni": számtalan ipari PC (strapabíróbbak és áruk közel duplája a házi használatra készülteknek), illetve mérőműszer tartalmaz pl. Intel 80188-ast, vagy 80386-os mikroprocesszort. A tervezett Marsjáróban pl. 8085-ös chip található. A Microsoft Windows 95-öt eredetileg 386-os CPU-ra készítették. Házi használat esetén szövegszerkesztésre és játszásra kiválóan megfelel egy 100 MHz-es 486-os konfiguráció, 16 MB RAM-mal, 500 MB HDD-vel és 4x-es CD-ROM meghajtóval.





Hordozható számítógép (palmtop)Ipari műszer 80188-as CPU-val
Windows CE operációs rendszerrel

A SZOFTVER

A számítógép megvásárlásával nem oldottunk még meg mindent, ez csupán a kiadás egyik (kisebbik) része. Meg kell szereznünk a gépet működtető programokat is:

- *választanunk kell* egy operációs rendszert (a piacon többféle is kapható),

amelyet fel kell telepíteni, utána installálni a különféle hardver kiegészítőket (pl. hangkártyát, modemet stb.) – egyes boltokban ezt a vásárlás előtt megteszik helyettünk,

- elő kell teremteni a legismertebb Commander, tömörítő és a legújabb víruskereső programokat (a kiadvány végén található bővebb leírás),
- meg kell szereznünk azokat a programokat, amellyel dolgozni fogunk (ha csak játszunk, akkor azokat),
- végül, de nem utolsó sorban néhány tíz márkás floppyt kell vásárolni, melyekre a legfontosabb anyagainkat archiváljuk és gondosan meg kell őriznünk a vásárláskor kapott install lemezeket is. Ha tudunk CD-t íratni, vegyünk üres írható CD-t (ára nagy tételben kb. 300 Ft/db) és az archiválást erre végezzük el.

Ezután kezdhetjük csak el a munkát. Használt számítógép vásárlásánál járjunk el óvatosan: a gépkocsikhoz hasonlóan rejtett hibákat tartalmazhatnak! A számítógépek ára világszerte évek óta rohamosan csökken, a magyar árak azonban nincsenek mindig szinkronban a nemzetköziekkel. Néhány alkatrész ára jóval olcsóbb, mint külföldön, de ennek fordítottja is elképzelhető.

Egy 80-as években írt régi program azért futtatható mostani PC-n is, mert a CPU ismeri a régebbi mikroprocesszorok üzemmódját és átkapcsol rá. Hasonlóan a képernyős megjelenítő egység is visszavált egy, már régen nem használt üzemmódra. A programoknak ilyen formán visszamenőleg való felhasználhatósága (felülről kompatibilitás) szintén növelte az IBM PC-k elterjedését.

A kompatibilitás sajnos sosem teljes körű: a különböző gyártók (Intel, AMD, Cyrix stb.) CPU-i 99%-ban kompatibilisek egymással. Vannak olyan szoftverek, amelyek csak javító program telepítése után futnak bizonyos gépeken. Sőt, az Intel CPU-kon belül a visszafelé (lefelé) kompatibilitás sem 100%-os.

2.3. Régebbi PC egységek

Már ritkán használt PC egységek felsorolásszerűen (a legrégebbivel kezdve):

- CPU: Intel 8086/8088, 80186/80188 (nem voltak PC-kben), 80286, 80386 SX/DX, 80486 SX/DX, Overdrive CPU-k, Pentium, Pentium Pro, Pentium MMX (MultiMedia eXtension). Opcionális tartozékuk volt a matematikai társprocesszor (co-

processzor), amelyet a 80486 DX-től egybeépítették a CPU tokjába.

- monitor és kártya: MDA, CGA, Hercules, EGA, VGA. A Monochome Display Adapter (MDA) csak szöveges üzemmódban működött, a többi szöveges és grafikus módban. Az SVGA kártya lefelé kompatíbilis: ismeri a CGA, EGA és VGA kártyák üzemmódjait.
- hajlékonylemez: 5.25"-os 360 KB vagy 1.2 MB kapacitású, 3.5"-os 720 KB kapacitású. (A " a lemez átmérőjét jelzi inch/col/hüvelykben: 1 inch = 2.54 cm.)

Érdekességképpen az 1981-es IBM PC felépítése: 4.77 MHz-es Intel 8088 CPU, 8 KB ROM-BIOS, 64 KB ROM-BASIC interpreter, 64 KB RAM, 2 db. 5.25"-es 360 KB-os floppy meghajtó, kazettás magnó csatlakozó. Operációs rendszere a Microsoft által a CP/M-ből kifejlesztett MS-DOS 1.0 vagy PC-DOS 1.0 volt: egy 360 KB-os lemezen elfért és nem ismerte a könyvtár fogalmát. 1982-ben jelent meg az IBM XT (eXtended Technology) max. 640 KB RAM-mal, 10 MB winchesterrel, MDA vagy CGA képernyő adapterrel. 1984-ben adták ki az IBM AT-t (Advanced Technology), melynek legfontosabb újdonsága a 6 MHz-es Intel 80286 CPU. Itt jelent meg először az akkumulátorral táplált belső óra és CMOS, 64 KB-os lett a ROM-BIOS és 1.2 MB a floppyk kapacitása. 1985-ben jelent meg a 32 bites 80386-os CPU.

2.4. A számítástechnika piac vezető vállalatai

A jelenlegi személyi számítógép világhoz három (amerikai) nagyvállalat neve fűződik:

- IBM – a moduláris felépítésű személyi számítógép (PC) megalkotója,
- Intel – az IBM PC-kben használt mikroprocesszor kifejlesztője,
- Microsoft – a leghíresebb, legelterjedtebb (és egyes körökben a legvitatottabb) operációs rendszerek (DOS, Windows) készítője.

A világ személyi számítógép állománya 80%-ban IBM PC kompatíbilis gép (a becslések szerint 150 millió darab), a maradék az Apple Macintosh, amely bizonyos tulajdonságaiban meghaladja az IBM PC-k jellemzőit, Amerikában pl. az oktatásban terjedt el. Az IBM PC-k kb. 90%-án a Microsoft valamelyik operációs rendszere fut, kisebb

részben Linuxot (ez egy PC-re írt, szabadon terjeszthető és felhasználható UNIX rendszer), IBM OS/2-t vagy egyéb operációs rendszert használnak.

2.5. Hol használják a számítógépet?

Szinte mindenhol! Ezért is fontos, hogy bizonyos szinten megtanuljuk használatát és ne idegenkedjünk tőle, mert rengeteg munkakör nélkülözhetetlen része lett.

A számítástechnika óriási ütemben fejlődik, amellyel csak *folyamatos tanulással* lehet többé-kevésbé lépést tartani. Nem elég tehát ezt a jegyzetet megtanulnunk és utána ülni babérjainkon, mert akkor tudásunk éppúgy, mint e jegyzet elavult lesz 2-3 év elteltével.

A kiterjedt használat miatt a végtermék néha hibát is tartalmazhat, amelyet észrevehetünk, ha nyitott szemmel járunk. Az alábbiakban pl. egy plakát részlete látható, amely 1999. tavaszán volt látható a budapesti mozikban:


PUSKIN

V., Kossuth L. u. 18.
TELEFON: 429-6080

Helyár: de: 350 Ft. du.+ este: 490 Ft. Kedd: PUSKIN NAP - egész nap 350 Ft. Pénztárnyitás hétköznap du.3. hétvégén de. f10. Este 7 óráig - kedd kivételével - az Ifjúsági és Nyugdíjas moziberletek érvényesek. Aktuális informácók: Error! Bookmark not defined. 3 légkondicionált terem. Dolby sztereo, Dolby digitális hang

Kiadványszerkesztési hiba
(előfordulhat, hiszen emberek vagyunk)

2.6. Y2K

Régen a háttértárak és a RAM memóriák kapacitása kicsi, áruk nagy volt. Emiatt a korai számítógépes programok és adatbázisok felépítése optimális, helytakarékos volt. Az árak zuhanásával az új szoftverek memóriaigénye is egyre nő. A kezdeti takarékosság egyik következménye a hamarosan bekövetkező ***2000. év problémája* (*"Y2K", "Year 2000 problem"*)**: a régi adatbázisokban az évszám mezőben nem tárolták el az évszázadot – az ezerkilencszázat -, így a forduló után, amikor is az évszám 99-ből átfordul 00-ra, problémák adódhatnak. Pl. ha egy banki program két naptári év különbségét kívánja ki-

számolni, negatív eredmény keletkezhet, vagy a 00-t nem 2000-nek, hanem – tévesen – 1900-nak veszik. Az Y2K-t a legújabb operációs rendszerek (pl. a Windows 98) helyesen kezelik, de *a probléma a programok forráskódjában található*, amit sajnos elég nehéz kijavítani, mert többnyire ma már régen nem használt nyelveken írták ezeket. A probléma tehát azokat a programokat érinti, amelyek időadatokat tárolnak, pláne amelyek műveleteket is végeznek velük. A megoldáson "gőzerővel" dolgoznak világszerte, a költségbecslések szerint milliárd USD nagyságrendű és a végeredmény korántsem biztos. További jóslásokba ne bocsátkozzunk, mert az esemény hamarosan bekövetkezik...



Amerikai újság Y2K karikatúrája

A piacon számtalan "csodaszoftver" kapható, amelyek letesztelik a számítógép összes egységét, beleértve az adatbázisokat is, kijelzik és kijavítják a hibákat. Egyébként 2038-ban a C++ nyelven írt programok egy részével lesznek hasonló problémák – a dátum visszaáll 1970-re -, szintén a használt dátumtárolási formátum miatt.

2.7. Legális, vagy illegális?

A számítástechnikai programok legálisan drágák, egy licensz akár millió Ft-ba is kerülhet. Ahogy az élet más területein, itt is lehetőségünk nyílik feketén – ingyen – beszerezni a különféle szoftvereket. A beszerzést elősegítette a CD író készülékek árának drasztikus csökkenése is. Pl. a Csillagok Háborúja első részét az amerikai megjele-

nés után két héttel már meg lehetett nézni Magyarországon is – Internetről letöltve, azaz a legújabb termékek is megszerezhetők szinte azonnal.

Az illegális szoftverkereskedelem ellen a nagy szoftvergyártók létrehozták a BSA-t (Business Software Alliance), amely harcol az illegális másolatok készítői, terjesztői és felhasználói ellen.





Orosz eredetű, nagyüzemben készült illegális CD és borítója 1998-ból

2.8. A számítógép és az ember kapcsolata (Kiemelten kiegészítő, vitaindító bekezdés!)

A számítógép működése a végletekig leegyszerűsítve abból áll, hogy a CPU "mechanikusan" végrehajtja a számára előírt gépi kódú programot, amely vagy a ROM-ban, vagy a RAM-ban található (utóbbi esetben a háttértárról, mint fájl töltődött be a RAM-ba). Mi a helyzet akkor, ha a gép előtt egy ember ül? Mind a számítógép, mind az ember elektromágneses hullámokat bocsát ki és használ. Bár nehezen hihető és az ilyen irányú kutatások még nem teljes körűek, de *elképzelhető*, hogy *bizonyos esetekben bizonyos emberek* képesek befolyásolni a számítógép működését, azaz a gép más ember közelében másképp működik (működhet). A helyzet hasonló a szóbeli vizsgához: a tételhúzásnál bármely tétel kihúzására elvileg egyforma valószínűségünk van, mégis akadnak olyan vizsgázók, akik ezt az egyenlő esélyt a számukra legkedvezőbb tétel irányába meg tudják változtatni.

Kit engedjünk gépünk elé?

A PC (Personal Computer), mint ahogy nevében benne van, *személyi* számítógép, elsősorban egy emberhez kötődik. Használata hasonlít az autóvezetéshez: sok szabályt kell *tudatosan* betartani ahhoz, hogy működőképes maradjon és rosszindulatúan egy pillanat alatt el lehet rontani. Az elrontás elsősorban programjaink és adataink letörlését jelenti, ami több napos többletmunkát jelenthet; hardver hibát jóval nehezebb okozni, de nem lehetetlen.

Azaz ha valakit gépünk elé engedünk, tejhatalmat adunk neki, bármit tehet! Ez az állítás a leggyakrabban használt operációs rendszerrel (MS-DOS, Windows 3.x, Windows 95/98) működő PC-kre vonatkozik. WIndows NT és UNIX esetén az egyes felhasználók jogait az ún. rendszergazda beállíthatja, így a vendég felhasználó rosszindulatú cselekedetei minimalizálhatók.

Hol a hiba?

Sajnos a számítógép használatakor fogunk találkozni az ún. *lefagyás*-sal (más neve kiakadás, lemerevedés, Windows alatt a képernyő színe miatt szokás még Nagy Kék Halál-nak is nevezni). Ilyenkor a kontrollt elveszítjük a gép felett és a gép nem hajlandó semmit sem csinálni, csak ki- és bekapcsolással éleszthető újra (erre mondják, hogy "végtelen ciklusba került"), vagy az újraindítás magától bekövetkezik. Mindkét eset az éppen végzett munka adatainak elvesztésével jár, ezért is fontos a gyakori mentés és archiválás.

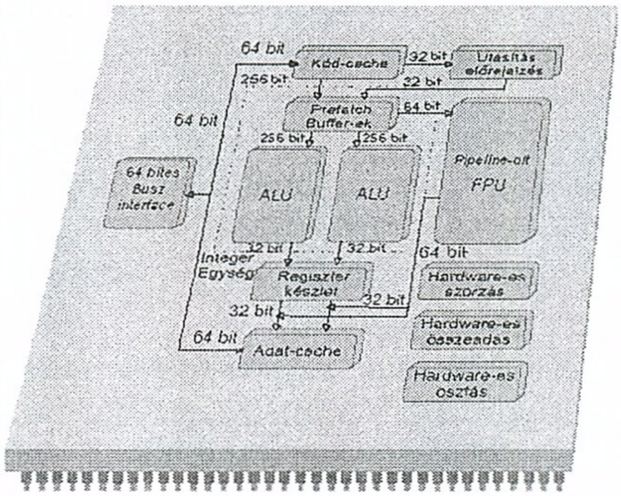
A hiba oka többféle lehet: leginkább a futó felhasználói program, vagy az operációs rendszer tartalmazza a hibát (angolul *bug*-ot). A hardverben általában megbízhatunk, ritkán mégis előfordulhat, hogy ott a probléma oka. Végül nagyon ritkán a számítógép környezetében bekövetkező elektromágneses zavarok (pl. villámlás) is módosíthatják a RAM tartalmát. A hibák többségével együtt kell élnünk (hacsak nem mi írtuk a használt programot), esetleg megpróbálhatjuk beszerezni a javított változatokat.

Példa a hardver hibára: a legelső Pentium CPU-k (az 1995. január 1. előtt készültek) koprocesszorában egy kis programozási hiba volt, amely nagyon csekély valószínűséggel a lebegőpontos osztásnál hibás eredményt adhatott. A hibáról szól az alábbi régi vicc – forrás: Internet -, amely szerint egy Pentium és egy 286-os versenyeznek:

- Mennyi 2+2? – kérdezi a 286-os.
- Öt!

- Nem is annyi!
- De gyors voltam, ugye?!

3. A Pentium mikroprocesszor felépítése



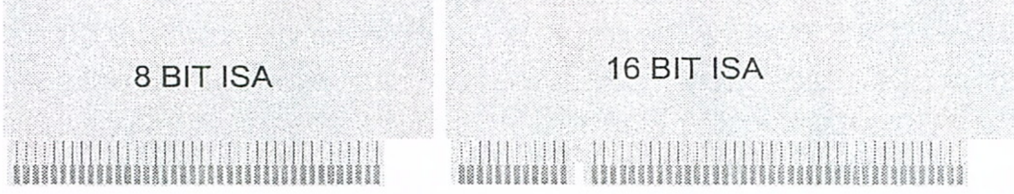
A Pentium mikroprocesszor funkcionális felépítése

- A Pentium egyik része, a **busz-csatolóegység** (BIU) adatokat és kódolt utasításokat kap a RAM-ból. A processzort a RAM-mal az alaplap egyik áramkörrendszere köti össze, az utóbbit nevezik busznak. Az adatbusz 64 bites, azaz az adatok 8 bájtosával kerülnek be a processzorba.
- A busz-csatolóegység két különböző útvonalon küldi az adatokat és utasításokat, mindkettő egyszerre 64 bitet tud továbbítani. Az egyik útvonal egy 8 KB méretű tárolóegységhez, a ***gyorsítótár***hoz (*cache*) vezet, amelyet az adatok ideiglenes tárolására használnak. A másik útvonal egy pontosan ugyanilyen gyorsítótárhoz vezet, ezt pedig annak az utasítássorozatnak a tárolására használják, amely közli a processzorral, mit tegyen az adatokkal. Az adatok és utasítások addig maradnak a két gyorsítótárban, amíg a mikroprocesszor más részei igényt tartanak rájuk. A Pentium MMX-ben 32 KB-os mind az adat-, mind az utasítás cache mérete.
- Mialatt az utasítássorozat a megfelelő gyorsítótárban várakozik, a mikroprocesszor egy másik része, az ágkiosztás-vezérlő egység átnézi az utasításokat, hogy eldöntse, a két aritmetikai-logikai egység (ALU) közül melyik tudja ezeket hatékonyabban kezelni. Az átnézés kiküszöböli az olyan helyzeteket, hogy az egyik ALU várakozni kényszerül, amíg a másik befejezi a végrehajtás alatt álló utasítássorozatot.

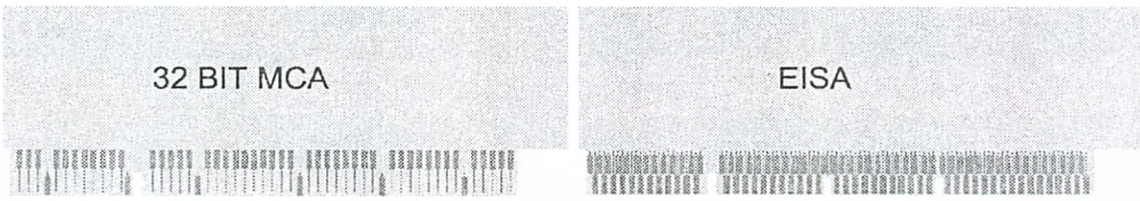
- A behívás előtti utasítástároló kiemeli az ágkiosztás-vezérlő egység által megjelölt programrészt, és a dekódoló egység a szoftver utasításait lefordítja olyan utasítássorozatra, amelyet az ALU-k megértenek.
- Ha lebegőpontos számokat – vagyis tizedestört-részt tartalmazó számokat, mint például 33,3 – kell kezelni, akkor ezek egy specializált belső processzorba, a lebegőpontos egységbe kerülnek.
- A végrehajtó egységen belül két aritmetikai-logikai egység dolgozza fel azokat az adatokat, amelyek kizárólag egész számokat tartalmaznak. Mindkét ALU egyszerre 32 bit mennyiségű utasítást tud fogadni az utasítás-dekódoló egységtől. A két ALU egyidejűleg dolgozza fel a saját utasításait. Ehhez azokat az adatokat használják fel, amelyek az adat-gyorsítótárból egyfajta elektronikus munkatárba, az úgynevezett *regiszter*ekbe kerültek.
- A két aritmetikai-logikai egység és a lebegőpontos egység munkájuk eredményét az adat-gyorsítótárba küldik. Az adatgyorsítótár a busz-csatolóegységbe továbbítja, amely viszont a RAM-ba küldi tovább az eredményeket.

4. Sínek

A számítógép külső eszközökhöz (idegen szóval *perifériák*hoz) való elektromos illesztésére szolgálnak a különféle csatlakozók. A *billentyűzet* és a tápegység saját dugaszon keresztül csatlakoztatható, míg a különféle perifériák (video-, I/O-, hálózati kártya, winchester-kontroller stb.) ún. *illesztősín*eken (*busz*okon) keresztül. A kártyákat az alaplapon található ún. *slot*okba kell dugni. A sín olyan párhuzamos vezetékekből álló, szabványosított összeköttetési rendszer, amelyen keresztül az egységek a mikroprocesszorral és egymással kommunikálnak.



A PC-k fejlődése során többféle rendszert használtak a számítógép és a processzor sebességi igényeinek megfelelően. A legegyszerűbb az ún. ISA busz (Industry Standard Architecture), amely állandó sebességű (8 MHz) aszinkron üzemmódban működik. Két változata van, a 8 bites (XT) és a 16 bites (AT). A 8 bitessel max. 1 MB memóriát lehetett megcímezni, 8 adatvezetékkel és 20 címvezetékkel rendelkezett. A 16 bitessel 16 MB memória címezhető meg, 16 adatvezetéke és 24 címvezetéke van ("AT busz").



A processzorok sebességének és adatszélességének növekedésével ez az állandó sebesség a gépek működési korlátjává vált. Az IBM 1987-ben a PS/2 sorozathoz kifejlesztette saját 32 bites aszinkron buszát, a mikrocsatornát (Micro Channel – MCA), mely nem volt kompatíbilis a korábbi AT buszokkal. A nagyvállalat nem adta át a licensz jogát a klóngyártóknak, ezért a többi gyártó 1989-ben létrehozta a szintén 32 bites buszt, az EISA-t (Extended ISA), mely felülről kompatíbilis volt az ISA-val és bárki felhasználhatta. Sebessége megegyezett az alaplapi oszcillátor órajelével, illetve programozható volt.

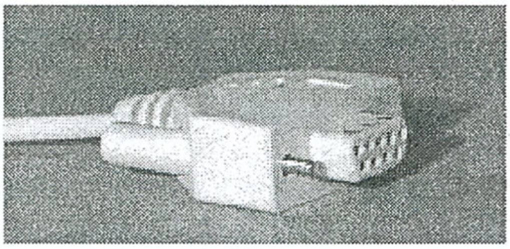


Ezt követte a mintegy 120, főleg távol-keleti gyártót tömörítő VESA bizottság (Video Equipment Standards Association) által kidolgozott lokális busz. A 32 bites VESA local busz (VL) kiegészíti az AT buszt, a behelyezett kártyák egyben az AT buszra is csatlakoznak. Saját vezérlőegységgel rendelkezik, órajel sebességű aszinkron kapcsolatot biztosít 32 bites adatátvitellel.

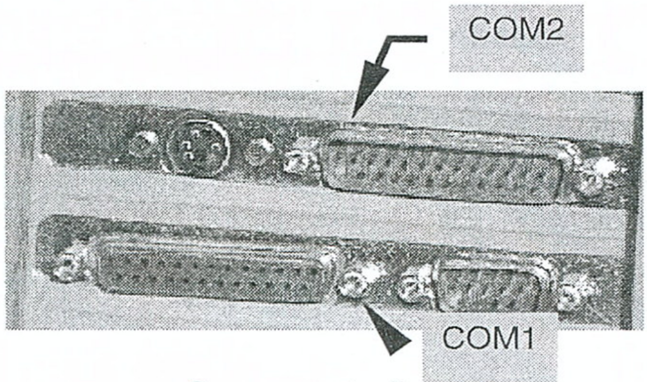
A ma használatos legfejlettebb megoldás az Intel által kidolgozott, állandó 33 MHz órajel sebességgel dolgozó 32 (386/486) vagy 64 bites (Pentium) PCI busz (Peripherial Component Interconnect). A PCI az AT busztól teljesen független, külön vezérlő áramkört igényel. Akárcsak a VESA, nem szabadalmaztatott, tehát nyitott architektúra.

5. Szabványos interfészek

5.1. Soros port



A soros port csatlakozója



Soros port aljzatok
a számítógép hátoldalán

Más neve Aszinkron, Serial, RS-232C, vagy COM port. Kialakítása ipari szabvány. Legfontosabb tulajdonsága, hogy az adatátvitel úgy történik, hogy *egy bájt bitjei időben egymás után haladnak át az adatvezetéken.* Az adatátvitel (5-6-7-8 bites adatok), sebessége (50-9600 Baud) és a paritásvédelem (páros-páratlan-nincs paritásvédelem) szoftverből állítható.

A soros portra legtöbbször az egeret csatlakoztatjuk, de modem is rákapcsolható, illetve két számítógépet is összeköthetünk segítségével (ún. 7 eres null-modem kábellel; a ”null modem” jelentése: a számítógép-modem-vezeték-modem-számítógép sorrendből a modemek kimaradnak, valamint az egyik végén ”adat ki” kivezetés a kábel másik végén ”adat be” és fordítva).

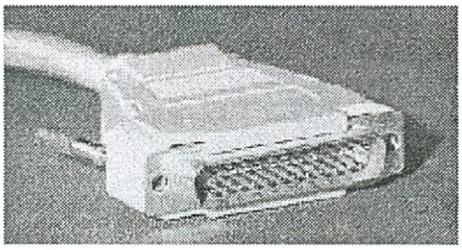
Az RS-232 csatlakozó kiosztása:

Csatlakozó száma	*A jel megnevezése*	*Adatáramlás iránya IBM PC külvilág*
1	föld	
2	adat ki	→
3	adat be	←
4	adás kérés	→
5	vételkészség nyugtázó jel	←
6	adatkommunikációs berendezés készenléti jel	←
7	jel föld	
8	adatterminál készenléti jel	→
9	hívás indikátor	←

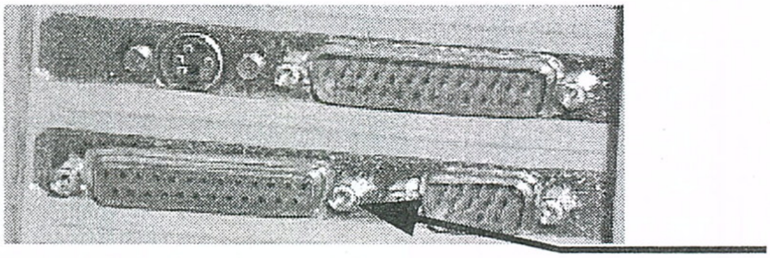
5.2. Párhuzamos port

Más neve Parallel, Printer, Centronics, vagy LPT port. Az adatátvitel úgy történik, hogy *az átküldendő bájt bitjei egyszerre, több adatvezetéken egyidőben kerülnek továbbításra.* Az adatátvitel így gyorsabb, viszont több vezetékre van szükség.

A párhuzamos portra legtöbbször a nyomtatót csatlakoztatjuk, ráköthető külső hordozható winchester is, vagy két számítógép összekötésére használhatjuk. Nem lehet összekeverni a soros porttal, mert a soros port aljzata "tüskés", a párhuzamosé pedig "lyukas".



A párhuzamos port csatlakozója



Párhuzamos port aljzata a számítógép hátoldalán

A párhuzamos port kivezetései:

Csatlakozó száma	*A jel megnevezése*	*Adatáramlás iránya IBM PC külvilág*
1	"adat érvényes"	→
2..9	adatvezetékek (0..7-es bitek)	→
10	adatvétel tényének nyugtázása	←
11	eszköz (nyomtató) foglalt	←
12	kifogyott a papír	←
13	a nyomtató működik (on/off line)	←
14	automatikus soremelés be/ki	→
15	(nyomtató) hibajelzés	←
16	(nyomtató) alaphelyzetbe állítása	→
17	a PC jelzi a nyomtatónak, hogy kész a küldött adatok fogadására	→
18..25	föld	

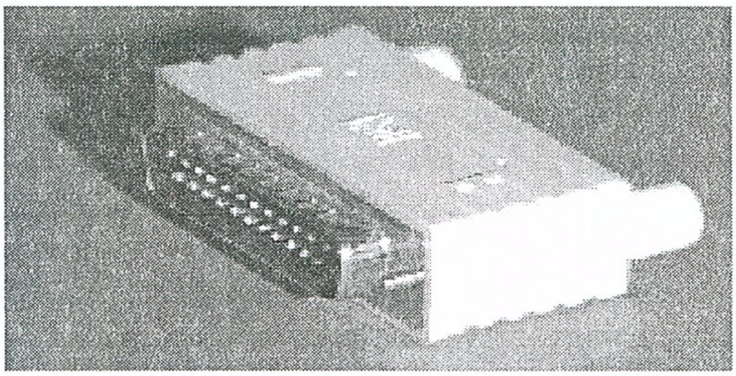
Két számítógép összekötése: a soros vagy párhuzamos kábel néhány száz forintért kapható a boltokban. A kapcsolat létrehozása után az egyik gép "látja" a másik háttértárait és portjait. Ez a lehetőség nagyon kényelmes, főleg nagy adatátvitelnél, mert sokkal gyorsabb, mintha floppy segítségével másolnánk. A kapcsolathoz a 6-os

MS-DOS `interlnk.exe/intersvr.exe` külső DOS parancsait használhatjuk, a Norton Commander "Link" opcióját, vagy a Windows 95/98 `directcc` programját. Hálózati kártya és számítógép-hálózat esetén a lehetőségek megtöbbszöröződnek, ez azonban már más témakör → lásd Számítógép hálózatok tantárgynál.

5.3. Game port

Botkormány (joystick) csatlakoztatható hozzá. Elsősorban játékprogramok használják.

5.4. SCSI



SCSI lezáró

Újabb fejlesztés a SCSI (Small Computer System Interface), mely egy második buszt hoz létre, 7 illetve 15 egységet kapcsolhatunk rá, a vezérlőn kívül. Mindegyik egység külön sorszámot kap, a legutolsó után egy lezárót (*terminátort*) kell alkalmazni. Nagy adatátviteli sebességű (5-20 MB/sec). Leginkább winchestert, CD írót, szkennert vagy DAT-ot csatlakoztatnak rá.

5.5. USB

Az **Universal Serial Bus**-ra egér, monitor, botkormány, modem, nyomtató, szkenner stb. csatlakoztatható. Nagy adatátviteli sebességű (12 Mbit/sec). A perifériákat a gép kikapcsolása nélkül lehet rá- és lecsatlakoztatni. Az USB automatikusan felismeri a csatlakoztatott eszközöket (*Plug and Play*) és újrakonfigurálja a PC-t. Összesen 127 egység csatlakoztatható a buszhoz.

6. Háttértárak

6.1. A merevlemez (winchester)

A ***merevlemezes tár***ak, amelyek kapacitása egyre növekszik, a mikroszámítógépek belső háttértárolói. Többféle változatát használják:

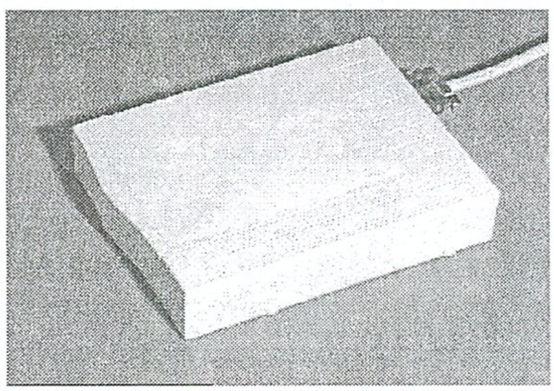
- beépített vagy külső csatlakozással (10 Mbyte – 9 Gbyte),
- cserélhető (40 Mbyte – 200 Mbyte),
- hordozható (20 Mbyte – 1 Gbyte).

A *merevlemezes tár*ban több mágnesezhető réteggel bevont könnyűfém lemez van zárt védőburkolatban elhelyezve. A lemezt ún. nano-technológiával készítik, hogy a szigorú alak- és mérethűséget biztosíthassák. A meghajtó 3600, vagy nagyobb fordulat/perc fordulatszámmal forgatja a merev lemezeket. Az állandó forgás miatt az olvasófejek nem érnek hozzá a lemez felületéhez, hanem ún. *repülő fej*eket használnak, amelyek a nagy kerületi sebesség következtében kialakuló légpárna hatására a lemez felületétől néhány μm távolságra kerülnek.

A képen látható, hogy mi van a *merevlemezes tár* dobozában:



Merevlemez meghajtó a fedél kinyitása után.
Többé nem használható...



9 GB-os külső winchester
Csak 8 GB (egyes kiírások szerint 8.388 GB) látható belőle.

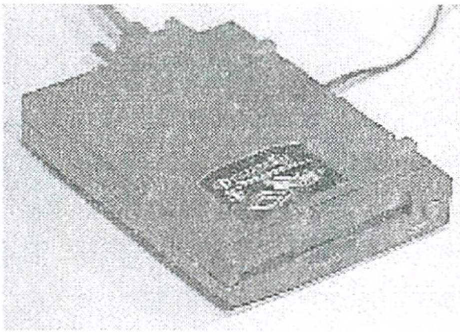
A mechanika lezárt fémházban található, így a por nem tud bejutni. Ha bejutna, a lemez és az író/olvasó fej közti keskeny résbe kerülve a lemez felületét megkarcolva annak meghibásodását okozná.

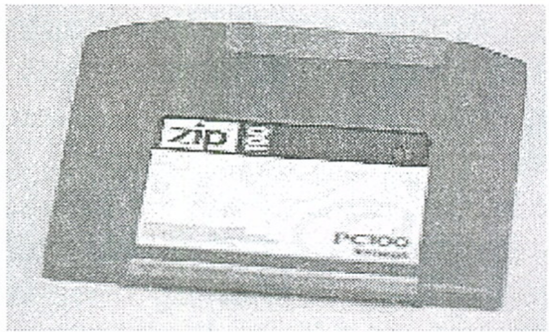
- Az egység alján található nyomtatott áramköri kártya az operációs rendszer által irányított lemezvezérlőtől – az ún. ***winchester kontroller***től – kapott adatokat átalakítja az író/olvasó fejeket mozgató parancsokká. Az állandó sebességgel forgó lemezről a kártya utasításai alapján a fejek adatokat olvasnak le, vagy írnak fel. Az IDE (Integrated Drive Electronics) meghajtóknál a kontroller a kártya része.
- A motorral hajtott tengely bizonyos számú, mágneses réteggel bevont lemezt forgat néhány ezres percenkénti sebességgel (a jelenlegi eszközökben általában 3600 ford/perc). A lemezek száma és a mágneses réteg kialakítása az egység kapacitásától függ. A jelenlegi meghajtókban a mágneses réteg vastagsága kb. 3 milliomod inch.
- A koncentrikus körökben elhelyezkedő ***sáv***ok (*track*) felett a fejeket egy kar mozgatja ki/be, nagy pontossággal. Az író-olvasó fejek kb. 150 nanométer, sőt még kisebb távolságra helyezkednek el a lemezek felett.
- A karra ráerősített fejek a megfelelő sávra (track) mozognak. Ezt követően adat írásakor a lemez felületén található mágneses részecskét átmágnesezik, olvasáskor a részecskék mágnesezettség irányát érzékelik.
- Amikor egy program fájl írását vagy olvasását kéri, az operációs rendszer a kontroller segítségével az író/olvasó fejeket a merevlemez fájl allokációs táblája (***File Allocation Table – FAT***) fölé mozgatja. A rendszer ennek olvasása után állapítja meg azt a helyet, ahol a létező fájl kezdődik, vagy ahol üres hely van új állomány létrehozására.
- Az operációs rendszer új fájl létrehozását a FAT-ban elsőnek megtalálható szabad helyen kezdi el. Előfordulhat, hogy a fájl nagyobb, mint amennyi üres terület van ott, ekkor egy másik területen folytatja az írást. Így egy-egy állomány rengeteg, különböző helyen található cluster-re kerül, szétszabdalva. A különböző részek pontos helyét a FAT láncolt formában tárolja.

A BIOS által elvileg kezelhető legnagyobb merevlemez mérete:

1024 sáv x 64 szektor x 256 fej x 512 bájt/szektor = 8 GB.

A BIOS megkerülésével (pl. Windows 98, FAT32) a fenti határ több ezer MB-ra nő!





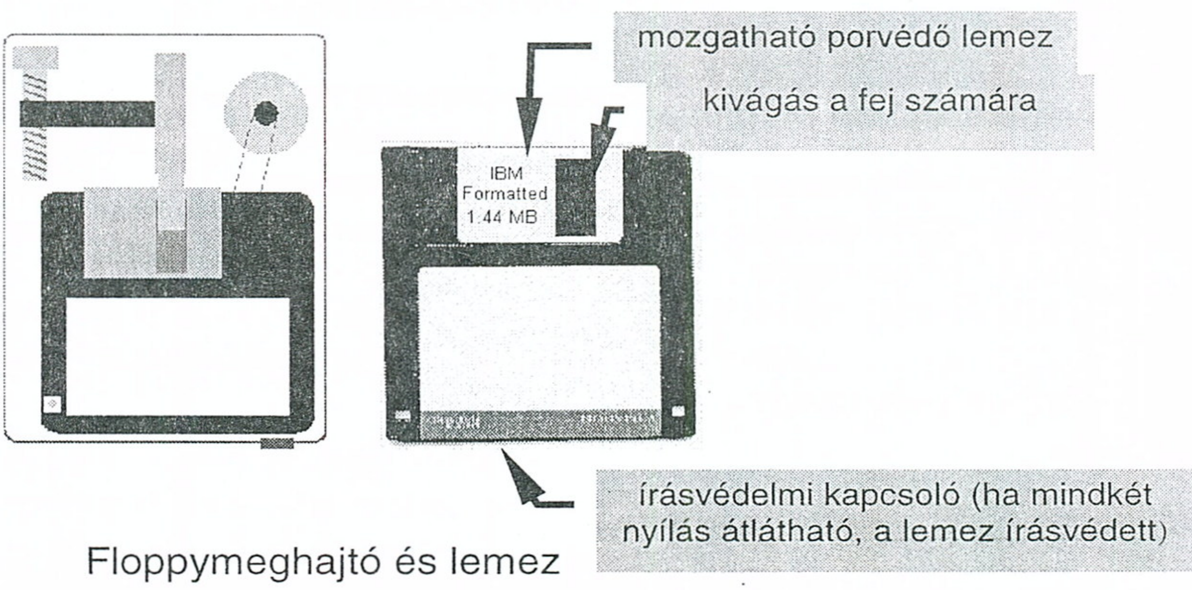
"ZIP drive": a párhuzamos portra köthető külső merevlemez-meghajtó. A cserélhető lemezek kapacitása 100 MB, újabban 250 MB. Az esetleges nyomtató az egység mögé csatlakoztatható.

6.2. A hajlékonylemez (floppy disk)

A hajlékonylemezes tár meghajtója a mikroszámítógép házába van beépítve, de a kör alakú mágneslemez, melyet műanyag burok véd az esetleges sérülésektől, kivehető, tehát az adathordozó üzemszerűen cserélhető. Másik előnyös tulajdonsága az alacsony ár (kb. 100 Ft/db). Sajnos hátrányai is vannak:

- aránylag kis kapacitású (IBM PC szabvány 1.44 MB, segédprogrammal max. 2 MB),
- sérülékeny és csekély megbízhatóságú (nagyon érzékeny az elektromágneses térre).

Az ábrán egy floppymeghajtó és egy hajlékonylemez látható:



Floppymeghajtó és lemez

• Amikor egy 3.5"-es floppyt berakunk a meghajtóba, mechanikusan érzékeli a lemezt, egy mechanizmussal félretolja a lemez mágneses rétegeit védő fémlemezt, ezáltal a lemez mindkét oldalára lehet írni illetve onnan olvasni. Az adatok a lemez felületén koncentrikus körök, *sáv*ok mentén helyezkednek el. A különböző típusú lemezek kapacitása a fizikai mérettől, a sávsűrűségtől és a rögzítési jelsűrűségtől függ. A jelenleg használt *nagy sűrűségű* (*High Density – HD*) floppyk 1.44 MB kapacitásúak. Régebben használták az SD-Single Density, DD-Double Density; SS-Single Sided (egyoldalas) lemezeket is, illetve az ún. "nagy floppyt", 5.25" inch átmérőjűt is (jellemző kapacitásaik: 360 KB, 720 KB, 1.2 MB voltak), mára már elavultak és a 3.5"-es HD-s floppy maradt.

• Más bütykök és görgők két író-olvasó fejet a lemez közvetlen közelébe visznek. A fejek két oldalról majdnem érintik a mágneses bevonatot. Mágneses impulzusokat bocsátanak ki, ezáltal megváltoztatják a bevonatban lévő mágneses részecskék polaritását.

• A meghajtó áramköri kártyája jeleket kap a vezérlő egységtől, melyet transzformál és előállítja az író-olvasó fejeket működtető villamos impulzusokat.

• Egy fénykibocsátó dióda (*Light Emitting Diode – LED*) átvilágítja az írásvédelemre szolgáló nyílást, ha az nyitott állapotban van. Ekkor a fényt egy fotodióda érzékeli és letiltja az írás műveletét.

• A motor egy körmös kapcsolóval forgatja (a fordulatszám kb. 360 fordulat/perc) a lemezt, amely a centrifugális erő hatására merevvé válik.

• Egy léptető motor, a vezérlő egységtől kapott villamos impulzusok hatására, bizonyos számú szögelfordulást végez jobbra vagy balra, s egy spirál alakú erőátviteli bordával ellátott tengely segítségével megfelelő pozícióba juttatja az író-olvasó fejeket.

• Amikor a fejek megfelelő helyzetbe kerülnek, akkor az író fejek kibocsátják a mágneses impulzusokat, vagy az olvasó fejek érzékelik a lemez részecskéinek mágneses polaritását.

A BIOS által elvileg kezelhető legnagyobb hajlékonylemez mérete:

256 sáv x 256 szektor x 2 fej x 512 bájt/szektor = 64 MB.

6.3. Streamer

Hagyományos hangkazettához hasonló tokba fűzött magas minőségű mágneses szalag. Nagykapacitású háttértár, használata kicsit nehézkes, időigényes. Biztonsági mentéseknél és archiválásokra használják.

6.4. DAT

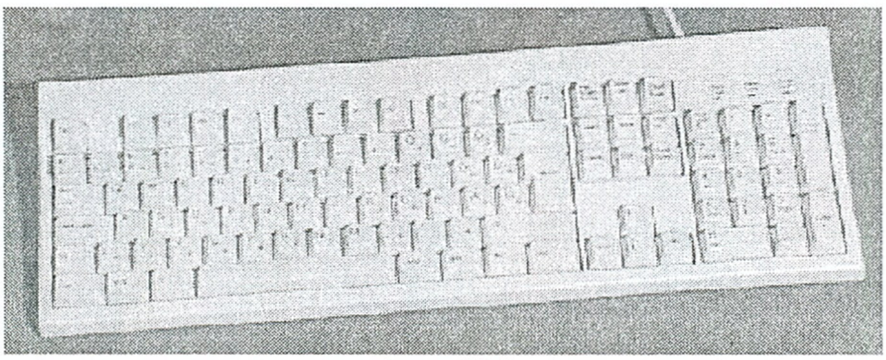
A Digital Audio Tape, digitális hangszalag rövidítése. Soros elérésű adathordozó. A kazetták tömörítetlen kapacitása eléri a 8 GB-ot.

7. Perifériák

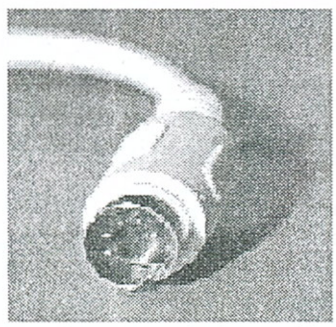
A perifériák különféle intelligens eszközök, amelyek a számítógép házán kívül találhatók és a gép hátulján, vagy oldalán található csatlakozóaljzatokon keresztül kapcsolhatók a rendszerhez.

A perifériák ún. *megszakítás*ok (*interrupt*) segítségével kommunikálnak a CPU-val: ha valamelyik egység a CPU-hoz kíván fordulni, a megfelelő vezetéken egy jelet küld. Amikor a CPU befejezi az aktuális utasítás végrehajtását és éppen ráér, "megszakítja" a futó program működését és "kiszolgálja a megszakítást". Az ún. nem tiltható, nem "maszkolható" megszakítás (*Non Maskable Interrupt – NMI*) a RAM, vagy a buszok paritáshibája esetén keletkezik.

7.1. Billentyűzet



105 gombos Windows 95-ös magyar billentyűzet...



... és csatlakozója

A billentyűzet (*keyboard*) a legfontosabb adatbeviteli egység (*standard input*). A PC billentyűzete külön egység, egy kábellel csatlakozik az alaplaphoz. Benne egy Intel 8049 vagy azzal kompatíbilis chip található, melynek feladata a billentyűzet kezelése. A chip másodpercenként 10–12-szer vizsgálja végig a billentyűzetet, egy gomb lenyomásakor egy ún. *scan-kód*ot küld a gép felé. A billentyű felengedését egy F0 hexa szám és a scan-kód jelzi. Ezeket az operációs

rendszer vezérlőjeleknek, vagy a beállítható kódlap szerinti karaktereknek értelmezi. Ha a billentyűt nyomva tartjuk, az egység egy idő után folyamatosan küldi a kódot (ismétlés).

Korábban az IBM PC/XT billentyűzete 84 gombot tartalmazott, napjainkra az AT mellett általánossá vált a 101 és a 102 gombos (vagy még több – lásd Windows 95) billentyűzet. Az 1993-ban kiadott magyar szabvány a 102 gombost fogadta el. A legfontosabb különbség, hogy az XT billentyűzete egyirányú kommunikációra volt képes, míg az AT-é kétirányúra.

A billentyűzet alapvetően három részre tagolódik. A legnagyobb, ún. ***alfanumerikus*** rész az írógépekre hasonlít. Itt találhatjuk meg az összes írásjelet, melyeket egyszerűen használhatunk. A profi (vakon gépelő) felhasználók számára az F és a J billentyűn külön kis kidudorodás is található az azonosítás megkönnyítésére. Az alfanumerikus részen néhány speciális billentyűt is találhatunk:

Enter (Return, kocsivissza) – az egyik legfontosabb billentyű. A beírt utasításokat, parancsokat tetszőlegesen alakíthatjuk, módosíthatjuk és az Enter hatására "véglegesítjük", ekkor küldjük el a számítógépnek végrehajtásra. Szövegszerkesztő programoknál a sor végének jelzésére szolgál, a Winword (Write, WordPad) esetében a bekezdés végét jelzi.

Esc – ez a különálló billentyű a legtöbb program esetében, ahogy elnevezése is mutatja (*escape – menekülés*) valamilyen befejezést, menüből való kilépést okoz. Másrészt az Esc egy olyan vezérlőkódot is jelent (a 27-est), amely egy kódsorozat első karaktere. Ilyen vezérlőszekvenciával (*Esc-szekvencia*) adhatunk parancsokat különböző perifériáknak, pl. a nyomtatónak.

Shift – váltóbillentyű, önmagában általában nincs jelentése. Az írógéphez hasonlóan betűkkel történő használatakor NAGYBETŰK jelennek meg, számok esetén a föléjük írt jelek (!, #, % stb.).

Ctrl (Control) – váltóbillentyű, önmagában általában nincs jelentése, nyomva tartása alatt egy másik billentyűt is le szoktak ütni. Pl. Ctrl+S – képernyő görgetés (*scroll*ozás) pillanatnyi megállítása, Ctrl+C – program befejezése (lásd még Pause/Break), vagy Ctrl+Alt+Delete – melegindítás.

Alt – a harmadik váltóbillentyű. A bal Alt-ot lenyomva a numerikus billentyűzeten egy 0-255 közötti számot írhatunk be, majd felengedésével ez a szám ASCII karakterként értelmeződik. Így olyan jeleket is bevihetünk, amelyek nincsenek a klaviatúrán (pl. nemzeti karakterek, amelyek 128 és 255 közöttiek: á=Alt+160, é=Alt+130 stb.). Másik szerepe a Ctrl-hoz hasonlóan billentyű-jelentésmódosítás (pl. Alt+X – Turbo Pascalnál kilépés DOS-ba stb.).

Tab (tabulátor) – szövegszerkesztési billentyű, segítségével a képernyőn egy soron belül nagyobb távolságokat ugorhatunk. A képernyő függőlegesen láthatatlan rácsokra van osztva, a Tab hatására a kurzor a következő rácspontba ugrik. Segítségével balra rendezett oszlopokat hozhatunk létre.

Caps Lock – kisbetű/nagybetű váltó. Megnyomva kisbetűk helyett nagybetűket használhatunk (ekkor a Shift+betű-re kisbetűket kapunk). Helyzetét egy LED jelzi (ha ég – NAGYBETŰS üzemmód).

Backspace (vagy balra mutató nyíl) – szövegbeírásnál a kurzortól balra álló karakter törlésére szolgál, egyben a kurzortól jobbra eső rész balra mozog.

A billentyűzet felső sorában 12 billentyű található F1..F12 jelöléssel. Ezek a gép funkcióbillentyűi. Jelentésük sokféle lehet, mindig az éppen használt programtól függenek (pl. F1-Help, F10 – menüsorba lépés stb.).

A klaviatúra jobb oldalán helyezkedik az ún. ***numerikus billentyűzet***. Itt található az összes számjegy, valamint a négy alapművelet, a tizedespont és az Enter megismételve. Hosszú számoszlopok beírásánál hasznos, az 5-ösön is elhelyeztek egy dudort a vakon írás megkönnyítésére. A számok beírása azonban csak akkor lehetséges, ha a NumLock billentyűhöz tartozó LED ég, egyébként a rájuk írt funkciók (Home, End stb.) aktiválódnak. A programok többségében mindegy, hogy egy szám, vagy művelet beírása honnan történik, néhány program azonban (pl. Norton Commander) másképpen értelmezi a numerikus rész billentyűit. Windows 95 alatt – beállítás után – lehetőség nyílik az egér irányítására (kurzormozgatás, kattintás stb.).

Végül az alfanumerikus és a numerikus rész között három csoportban néhány szürke billentyűt is találhatunk:

Insert – szövegszerkesztő programokban a beszúrás (Insert) vagy felülírás (Overwrite) üzemmód között lehet választani. Beszúrásnál az új karakter a kurzor pozíciójába kerül és a szöveg további része jobbra tolódik, felülírásnál a kurzor "feletti" karakter kicserélődik a leütöttel.

Delete – szövegszerkesztő programokban a kurzor "felett" álló karakter törlődik és a sor többi tagja balra csúszik (azaz egy Delete gomb helyettesíthető egy kurzor jobbra és egy Backspace-szel). Egyes számítógéptípusokon bekapcsolás után a Delete billentyűvel léphetünk be az ún. Setup-ba.

Home/End – valamilyen egység (pl. sor, képernyő, lista) elejére illetve végére helyezi a kurzort.

PageUp/PageDown – ahol használható, ott lapozni lehet a képernyőn felfelé illetve lefelé egy oldalt.

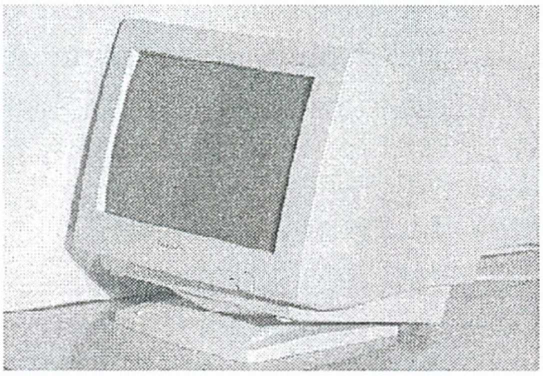
Kurzornyilak ($\rightarrow$ $\leftarrow$ $\uparrow$ $\downarrow$) – a képernyőn a kurzor, menüsorok, egér vagy más objektum mozgatására szolgáló, nagyon gyakran használt billentyűk.

PrintScreen – (egyes régi gépeken Shift+PrintScreen) a képernyő aktuális tartalmát kinyomtatja, nyomtató hiánya vagy egyéb hiba esetén hangjelzést ad. Grafikus üzemmód esetén sikeres eredményt akkor kaphatunk, ha előtte elindítjuk a `GRAPHICS.COM` programot. Windows alatt más hatása van: a képernyő tartalma (`.BMP` grafikus formátumban) a Vágólapra (Clipboard) kerül, Alt+PrintScreen esetén az aktív ablak tartalma kerül a Vágólapra. A billentyű másik jelentése (SysRq – System Request) eredetileg többfolyamatos rendszerben folyamatváltásra szolgált, ma nem használják.

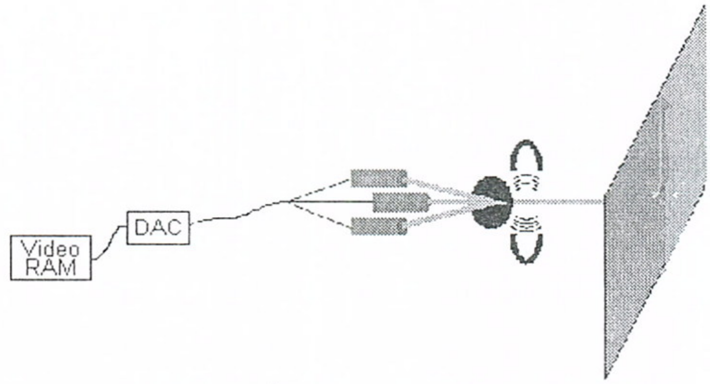
ScrollLock – ezt a nagyon fontos billentyűt a legritkább esetben használják, néhány régi játékprogramban a zene ki- és bekapcsolására szolgált. Egyes táblázatkezelőknél van funkciója.

Pause/Break – a Pause gomb megnyomására a program futása leáll és csak egy billentyű leütésére folytatódik. A Ctrl+Break (vagy Ctrl+C) megállítja a futó programot és a vezérlés visszakerül az operációs rendszerhez. Egyes programok ezeket a funkciókat le tudják tiltani.

7.2. Monitor



Monitor...



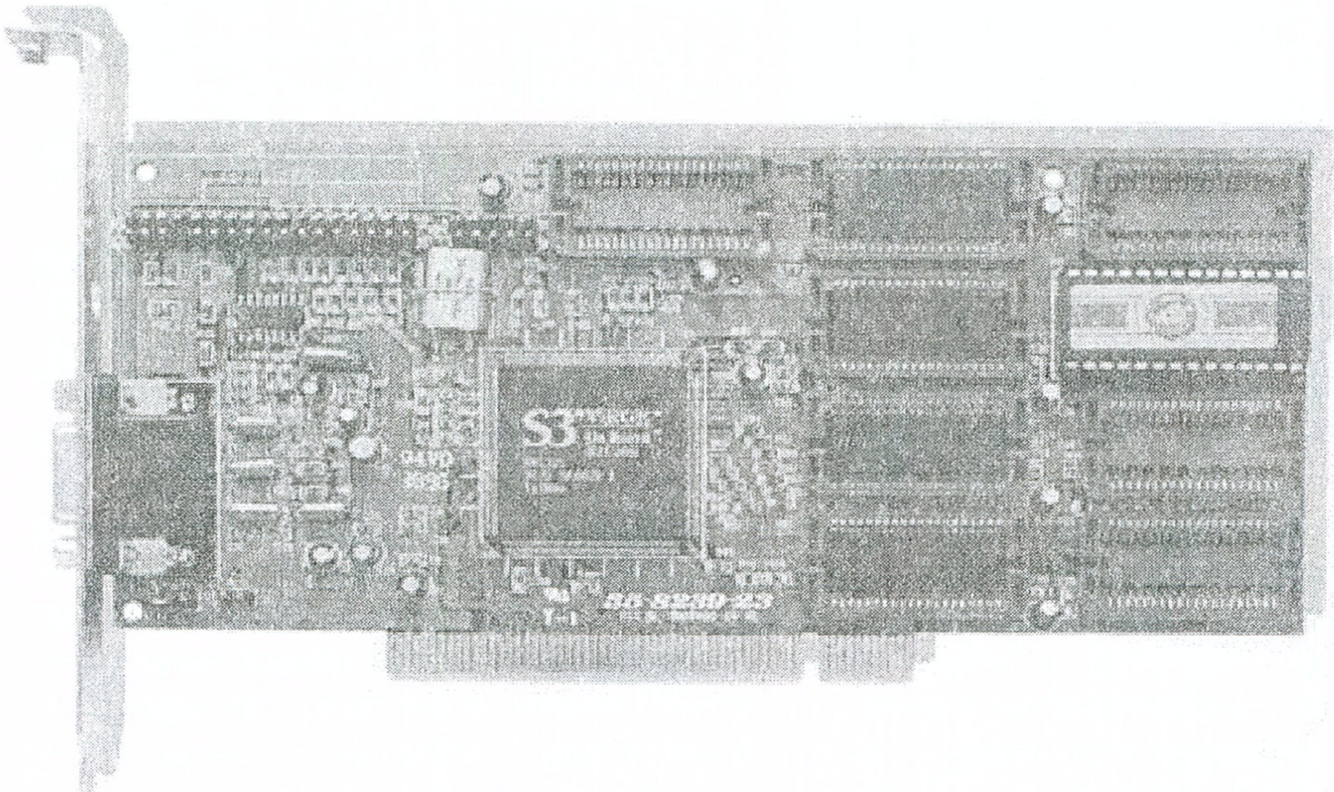
... és működési elve

A monitor a legfontosabb adatkimeneti egység (*standard output*). A legrégebbi számítógépek egy írógéphez hasonló ún. *teletype*-on jelenítették meg az eredményeket. A monitorok később jelentek meg. Közülük is először a *monokróm*, mely csak szöveget tudott megjeleníteni egy színnel (a másik szín a háttér volt). A grafikus és színes megjelenítők később alakultak ki.

Legfontosabb egységük a katódsugárcső. Csúcsában három elektronágyú helyezkedik el, három elektronsugár rajzolja a képet. A sugarak együtt haladnak, de mindegyik csak a saját elemi hálópontját világítja meg: a "vörös" sugár vörös foszfort, a zöld zöldet, a kék kéket (pontosabban foszfor és egyéb elemekből készült réteget.). A képernyő egy *elemi monitorpont*ját a háló megfelelő vörös-zöld-kék (Red-Green-Blue – RGB) foszforpontjai alkotják. A foszforpontok elhelyezkedhetnek háromszög alakban (delta képcső), vagy egymás mellett (Sony Trinitron). A foszfor a gerjesztési energia hatására fényt bocsát ki. A színek előállítása additív módon történik, e három alapszínből bármely szín kikeverhető (a szubsztraktív színkeverést használják pl. a különböző színű festékek keverésénél). A ***pixel*** (*képpont*) a képernyő legkisebb olyan egysége, amelyet a számítógép kezelni tud, monitortól függően egy vagy több elemi pontból áll. Az elektronsugarak a bal felsőtől a jobb alsó sarokig másodpercenként több tízszer végigpásztázzák a képernyőt.

A monitoron megjelenő kép leírása az alaplapra rácsatlakoztatható videokártyán található. Ez egy néhány MB méretű RAM ("video RAM"), amelyet a számítógépes programok el tudnak érni. Szöveges üzemmód esetén az igényelt memória kicsi: a karakterek ASCII kódjukkal és egy ún. *attribútum* bájttal (amely megmondja az előtér és a

háttér színét, az előtér intenzitását és hogy a karakter villogjon-e) tárolódnak, egy pozícióhoz tehát 2 bájt tartozik. Grafikus üzemmód esetén a kép képpontonként tárolódik a képernyőmemóriában, üzemmódtól függően néhány tíz KB-tól több MB helyet igényelve. A ma használt monitorok nagyobb felbontást biztosítanak a televíziókészülékeknél.

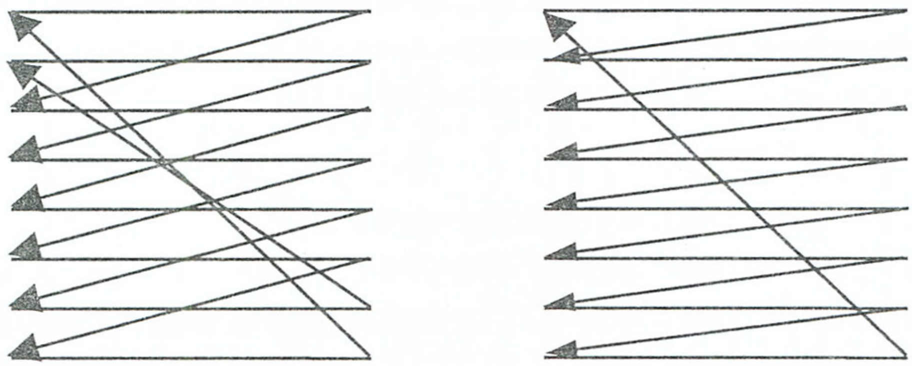


Videokártya

Szöveges üzemmód esetén a grafikus kártya először a betűk sorozatát bitmintázattá alakítja, az így kapott információt a képváltási frekvencia ütemében egy digitális-analóg konverter (DAC) analóg RGB videojellé alakítja. Ez azután a monitoron megjelenik.

Grafikus üzemmódnál nem egyszerű soros kódkonverziót kell végezni. A képernyő tartalmát darabokban – sokszor nem is sorrendben – kapja a grafikus kártya. A megjelenítés folyamatát a VGA kártyán levő processzor meggyorsíthatja. Ettől kezdve az összeállított kép megjelenítési folyamata már közös:

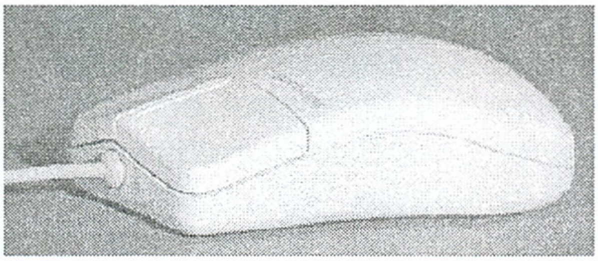
- Az analóg videojel alapszínenként (RGB) egy-egy elektronágyú intenzitását vezéreli, megadva ezzel a pillanatnyi képpont fényességét.
- Egy vízszintes és függőleges mágneses eltérítő rendszer a grafikus kártya által, a felbontástól és a képfrissítő frekvenciától függő-szinkronjel ütemével végigvezeti az elektronsugarakat a képernyő felületén. A képfrissítés frekvenciája az emberi szem korlátai miatt min. 50 Hz lehet, de manapság elérheti a 100 Hz-et is.
- A sugarak először egy fém maszkon haladnak át, amely biztosítja, hogy a megfelelő sugarak csak a saját színükhöz tartozó pontot villantsák fel.
- A képernyő *belső felületén* a három alapszínnek megfelelő foszfor pontok vannak, amelyek a becsapódó elektronsugár hatására felvillannak, így alakul ki a kép.
- A vízszintes eltérítés módja a felhasznált monitor frekvencia tartományától függően a nagyobb felbontási tartományokban (1024 x 768 felett) kétféle lehet:
 - jobb minőségű monitoroknál a elektronsugár minden képfrissítéskor az összes soron végigfut. Ez az ún. *teljes képfrissítés (non interlace)*,
 - ha a vízszintes eltérítés frekvenciája korlátozott, *váltósoros* (*interlace*) módot alkalmaznak. Ilyenkor az elektronsugár először minden páros, majd minden páratlan sort villant fel. Így egy teljes kép megjelenítése két ciklust vesz igénybe. Eközben a felvillanás erőssége a hosszabb idő miatt lecsökken, így a kép különösen kisebb képfrissítő frekvenciák esetén (50-60 Hz) kissé villogni fog. Ez a szemet, különösen közelről nézve fáraszthatja. A magyarországi televízióadások ezt a rendszert használják (50 Hz → másodpercenként 50 félkép jelenik meg).



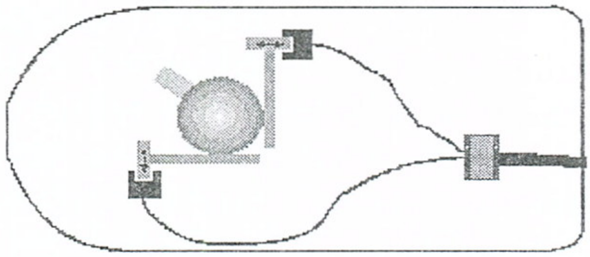
Magát a grafikát a rajzolóprogramok képpontonként (pixelgrafikus tárolás), vagy vektorok segítségével (vektorgrafikus rajzok) írják le. A képen található színek tárolása is többféle módon történhet: kisebb memóriát igényel, ha a kép csak 16 vagy 256 színt tartalmazhat. Ekkor az egyes képpontok a lehetséges színek sorszámát tartalmazzák (azaz 16 szín esetén egy képpontot 4, 256 szín esetén 8 bit ír le), az egyes színeket pedig az ún. ***paletta*** tartalmazza. A legvalósághűbb megjelenítés a nagy memóriaigényű ún. ***True Color*** (vagy *24 bites kép*), melynél minden képpont tárolására 3 bájt szolgál: egy-egy bájt tárolja a vörös-zöld-kék színösszetevők értékét. Ebben az esetben minden egyes képpont 256 x 256 x 256 = 16.7 millió különböző színt vehet fel.

Ha a monitoron sokáig ugyanaz a kép jelenik meg, egy idő után "beég" és halványan mindig látható lesz. Ezért találták ki az ún. ***képernyőkímélő programokat***. A fejlettebb monitorok, ha egy ideig nem nyúlunk a géphez, kikapcsolnak.

7.3. Egér



Optomechanikus egér kívülről ...



... belülről

Az interaktív programok képernyőmenüinek kezeléséhez egy kurzort kell a képernyőn mozgatnunk. Ahhoz, hogy ezt elvégezzük, olyan mechanikus eszközre van szükség, amely kezünk elmozdulását a számítógép által értelmezhető digitális jellé alakítja. Többféle ún. ***mutató eszköz*** (*pointing device*) terjedt el, a leggyakoribb közülük az *egér* (*mouse*) de van még ún. hanyattegér – trackball -, érintőtábla – touchpad -, érintőképernyő – touchscreen – és más megoldások is. Az egeret a 70-es években fejlesztették ki a XEROX laboratóriumaiban, mint számítógép szériatartozék az 1983-as Apple gépeken jelent meg először. A mai személyi számítógépekben használata nélkülözhetetlen.

Az egeret egy vízszintes felületen (az egérpadon – mouse pad) mozgathatjuk és a tetején található gombokat lehet megnyomni. Az Apple Macintosh-hoz egygombos egér tartozik, a Microsoft Windows kétgombos egeret igényel, más programok (pl. egyes CAD szoftverek) háromgombos egérrel működnek.

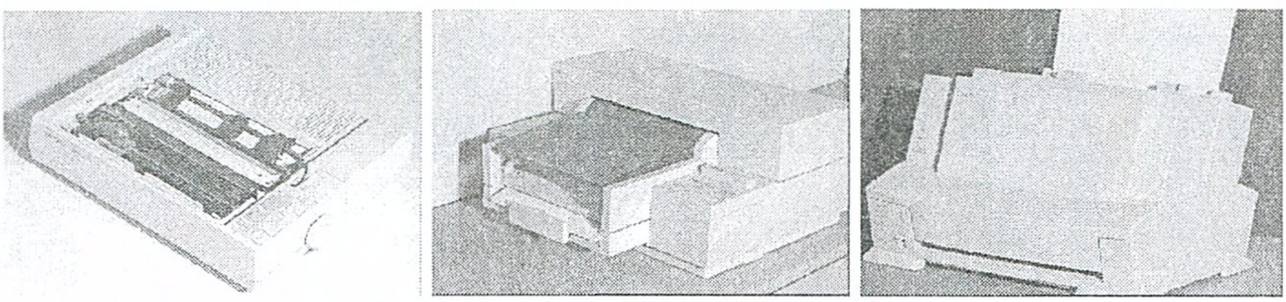
A legelterjedtebb – optomechanikus – egér működése:

- Az egér mozgatásakor tulajdonképpen az aljára erősített gumibevonatú golyót görgetünk (belül többnyire fém van). A golyóra két, egymásra merőleges műanyag görgő csatlakozik, amelyek a síkbeli elmozdulást vízszintes és függőleges összetevőre bontják fel.
- A görgőkkel egy tengelyen egy-egy "lyukas" tárcsa helyezkedik el, amelyek a görgőkkel együtt forognak. A tárcsa egyik oldalán LED, másik oldalán fényérzékelő fototranzisztor található. Az érzékelők előtt elhaladva minden lyuk (szegmens) egy impulzust generál. A keletkező impulzusok száma arányos az elmozdulással. Így az egér x és y irányú elmozdulását megkapjuk, előjeles egész számok formájában.
- A keletkezett impulzusokat bináris számmá alakítják.

- Az egér – mint periféria – megszakításon keresztül értesíti a számítógépet, ha elmozdulás történt. Az egeret kezelő felhasználói program időnként az egér meghajtó szoftverén keresztül lekérdezi a küldött jeleket és ennek megfelelően irányítja a kurzort a képernyőn. A kapott értékeket egy arányossági tényezővel – az egér érzékenység (*sensitivity*) – is megszorozzák, így kapják meg az elmozdulás nagyságát.
- Ha lenyomjuk az egér valamelyik gombját. a megfelelő bináris kód is továbbítódik a számítógéphez. A gombhoz tartozó eseményt (parancsot) már a felhasználói program hozza létre.

Az optikai egér nem tartalmaz mozgó alkatrészt. Egy négyzetrács felett kell mozgatni, amit az egér alján egy LED világít meg, így érzékelve az elmozdulást. Vannak új fejlesztések is, pl. IntelliMouse.

7.4. Nyomtató



Mátrix-, tintasugaras- és lézernyomtató. A legutóbbi nyomtatási sebessége percenként 5 A/4-es fekete-fehér oldal.

A **nyomtatók** a számítógépes programok eredményeinek papíron való dokumentálására szolgáló eszközök.

Ezek közül a karakterenkénti nyomtatási mód minden nyomtatónál használható, de a korszerű eszközök lapnyomtató módszerrel működnek. Ez utóbbiakhoz azonban a nyomtatóba egy nagysebességű processzort és megfelelő méretű (1-16 MB) RAM-ot építenek a feladatok végrehajtására.

A *mátrixnyomtatók* feje általában 9 vagy 24 tűt tartalmaz. Miközben a nyomtatófej vízszintesen mozog a papírlap előtt, egy helyi mágneses teret keltő elektromos impulzus a megfelelő tűket kimozdítja, amelyek a festékszalagon keresztül ráütnek a papírra.

A nyomtatott eredmény így keletkezik, minősége nem a legjobb (beállítható pl. *Near Letter Quality – NLQ*). A mátrixnyomtatókat a személyi számítógépeknél csak speciális esetekben használják – pl. szállodákban számlák nyomtatására -, felhasználási körük inkább nagygépes rendszereknél üzemeltetési adatok nyomtatása (pár száz oldalas percenkénti többpéldányos nyomtatásra a mátrixnyomtató kiválóan alkalmas és megbízható működésű) Előnyük tehát a többpéldányos nyomtatás és a megbízhatóság, hátrányuk a minőség és a zaj.

A *tintasugaras nyomtatóknál* nincs fizikai érintkezés a papír és a fej között. A karaktereket és rajzokat tintacseppecskékből álló sugár kilövésével állítják elő. A végeredmény minősége sokkal jobb, mint a mátrixnyomtatóé.

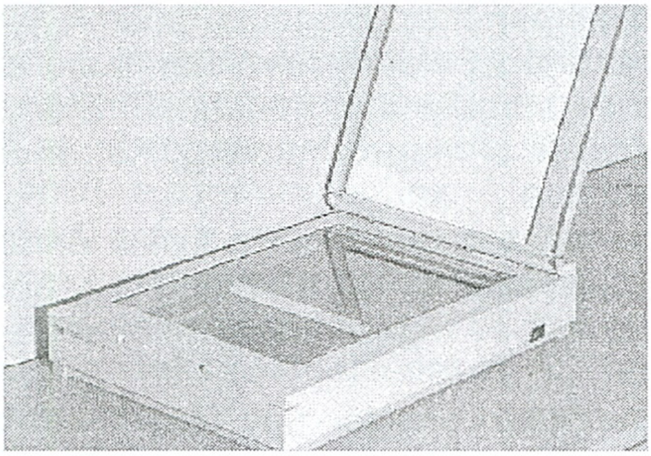
A *lézernyomtatókban* először egy fényelektromos vezető henger felületét negatív töltésekkel egyenletesen feltöltik. Utána szelektív megvilágításnak vetik alá a hengert, a megvilágított részeken a henger elveszíti negatív töltését és kialakul rajta a nyomtatandó rajz virtuális elektrosztatikus képe. A megvilágítást lézersugár, diódasor vagy halogénlámpa végzi. A megvilágítás befejeztével a henger érintkezésbe lép a finom szemcséjű gyanta-festék keverékkel, amely elektrosztatikusan szintén töltött, de ellenkező előjellel. A festék szemcséi hozzátapadnak a henger felületének ellentétes töltésű részeihez – láthatóvá válik a kép. Végül a kép átkerül az előzőleg pozitív töltéssel feltöltött papírlapra, majd a festék rögzítése érdekében a papír két henger között halad át.

Régebben használták a hőre érzékeny papírral működő *hőnyomtatókat*. A nyomtató nem használ sem festéket, sem szalagot, csupán a nyomtatófejbe épített ellenállásokkal dolgozott. A papír fény hatására lassan tönkremegy. Jelenleg speciális helyeken használják.

A nyomtatás feltételei a nyomtató oldaláról (sokszor megfeledkezünk valamelyik pontról):

- legyen hozzácsatlakoztatva a számítógéphez,
- legyen bekapcsolva,
- az sem árt, ha van benne papír,
- ha a nyomtatón van ON-LINE gomb, be kell kapcsolni (a hozzá tartozó LED világítson). Kikapcsolt állásban (OFF-LINE), a papírt fűzhetjük ki-be, de nyomtatni nem tudunk.

7.5. Lapolvasó (scanner)



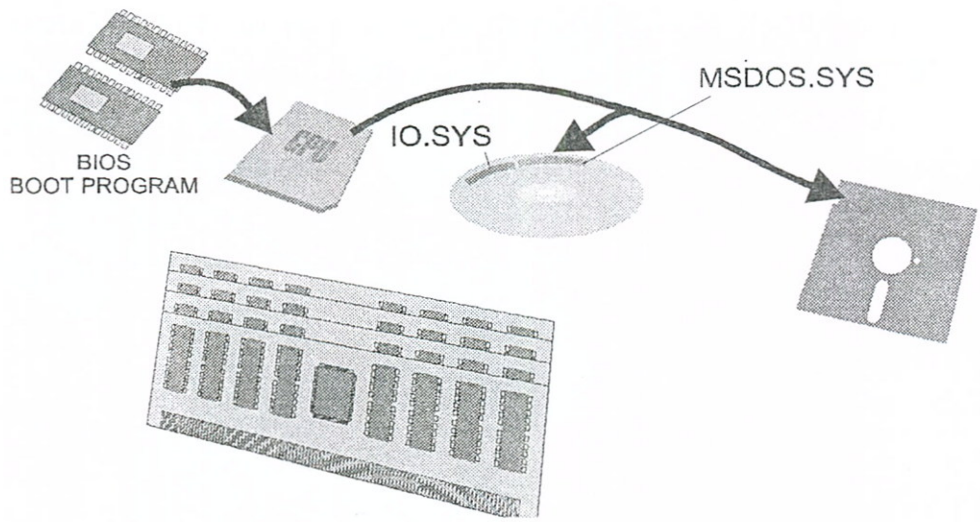
Scanner

A szkenner fekete-fehér, vagy színes képek bevitelére szolgál. A beolvasott adatok egy grafikus feldolgozó programba kerülnek, ezután az eredményt egy szokásos grafikus formátumban elmenthetjük, módosíthatjuk.

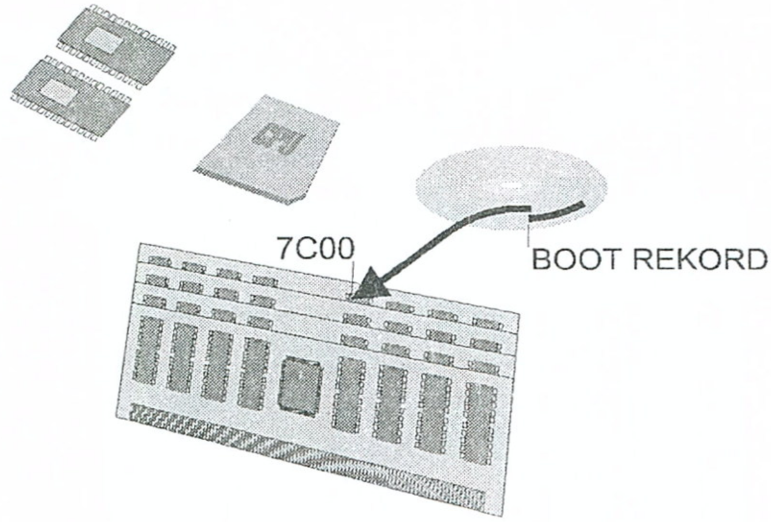
A szkennerek és a nyomtatók legfontosabb jellemzője a ***felbontóképesség***. Pl. 300 dpi → *dot per inch*, 1 inchen 300 (kép)pontot tud elkülöníteni. A másik adat a *színmélység*, amely azt mondja meg, hogy egy (kép)pont hány színű lehet. Csökkenő ára miatt – pl. Amerikában 50 USD-ért is kapható – egyre jobban elterjed. Létezik kézi, illetve asztali változata.

8. Az operációs rendszer betöltése

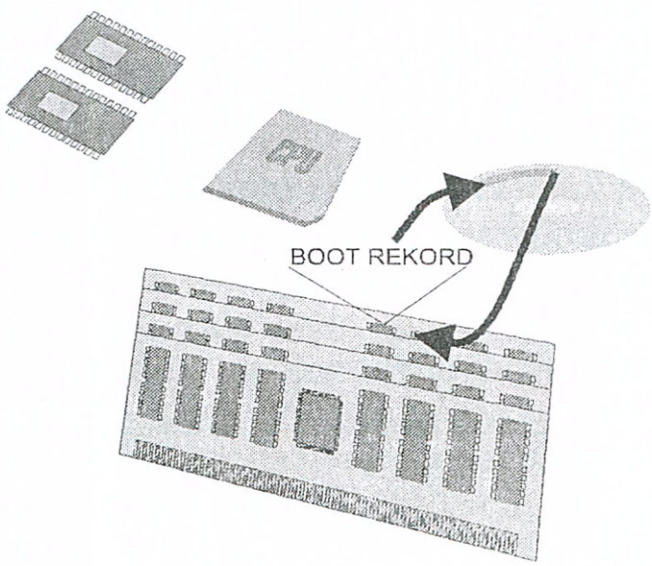
Bemutató példaként nézzük meg a már egyre ritkábban használt MS-DOS operációs rendszer betöltésének – az ún. *boot*olásnak – folyamatát.



- Amikor bekapcsoljuk a számítógépet, a ROM-BIOS ellenőrzi az összes hardver komponenst, majd a rendszerbetöltő (boot) program megnézi, van-e az A: lemezmeghajtóban formázott hajlékonylemez. Ha igen, akkor a program a lemez előre megadott pontjain keresni kezdi azokat a fájlokat, amelyek az operációs rendszer első két részét alkotják. Ezeket a rendszerfájlokat általában nem látjuk, mert mindegyiküknek olyan fájl-attribútuma van, amely elrejti őket a DOS `dir` parancsa elől. A Microsoft DOS-változatában (MS-DOS) ezeknek a fájloknak a neve `IO.SYS` és `MSDOS.SYS`, az IBM DOS-változatában (PC-DOS) pedig `IBMBIO.COM` és `IBMDOS.COM`. Ha a hajlékonylemez-meghajtó üres, a rendszerbetöltő program a C: merevlemezen keresi a rendszerfájlokat. Ha a rendszerbetöltésre kijelölt lemez nem tartalmazza ezeket a fájlokat, a program a következő hibaüzenetet küldi ki: *Non-system disk or disk error. Replace and strike any key when ready* (*Nem rendszerlemez vagy lemezhiba. Cserélje ki, majd nyomja meg bármelyik billentyűt*). Az esetek túlnyomó többségében ennek az az oka, hogy egy nem rendszerbetöltő (nem "bootolható") hajlékonylemez van a meghajtóban. Ha ezt a lemezt kivesszük és megnyomjuk bármelyik billentyűt, a rendszerbetöltés rendben végbemegy. Ha viszont üres hajlékonylemez-meghajtónál kapjuk ezt az üzenetet, ennek komolyabb oka van (általában az, hogy a winchesteren nincs operációs rendszer, mert *véletlenül* letöröltük, vagy vírus van jelen stb.).

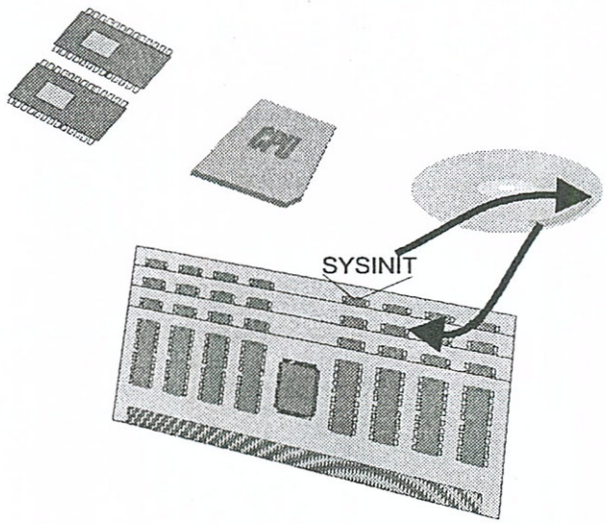


• Amikor a rendszerbetöltő program megtalálta a rendszerfájlokat tartalmazó lemezt, akkor beolvassa a lemez első szektorában található adatokat, és elhelyezi azokat a RAM meghatározott területén. Ez az adattömeg alkotja a DOS rendszerbetöltési (boot) rekordját. A boot rekord minden formázott lemezen ugyanazon a helyen található. Hossza alig 512 bájt, éppen elég arra, hogy a két rejtett rendszerfájl betöltéséhez szükséges utasításokat tartalmazza. Miután a BIOS rendszerbetöltő programja elhelyezte a boot rekordot a 7C00 hexadecimális RAM címen, egy ugró utasítással átadja a vezérlést.

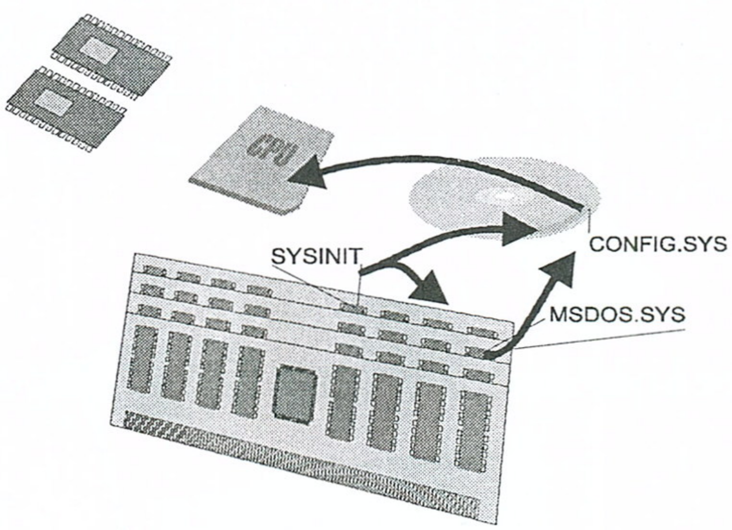


• A boot rekord átveszi a számítógép irányítását, és betölti az `IO.SYS` fájlt a RAM-ba. Ez a fájl kiegészíti a ROM-BIOS -t, és tartal-

maz egy SYSINIT nevű rutint, amely a rendszerindítás hátralevő részét irányítja. Az **IO.SYS** betöltése után a boot rekord tartalmára már nincs szükség, így ennek RAM-beli példánya nyugodtan felülírható.

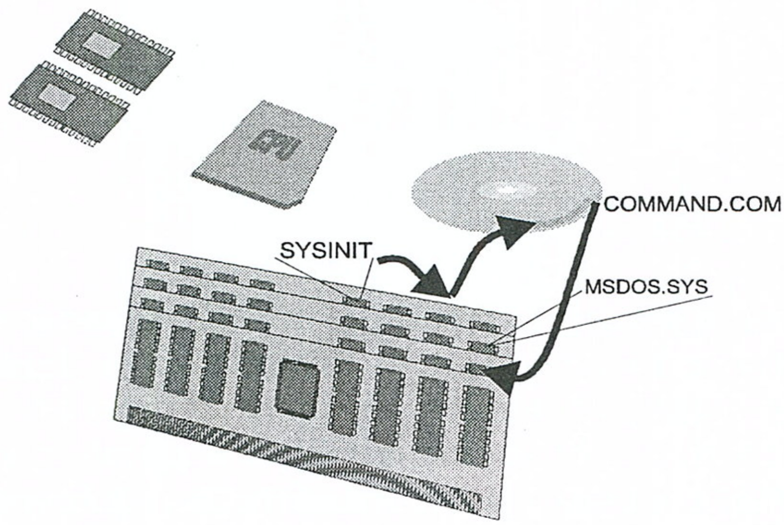


- A SYSINIT átveszi a rendszerindítás irányítását, és az **MSDOS.SYS** fájlt betölti a RAM-ba. Az utóbbi a BIOS-szal együttműködve kezeli a fájlokat, végrehajtja a programokat, és válaszol a hardvertől érkező jelzésekre.

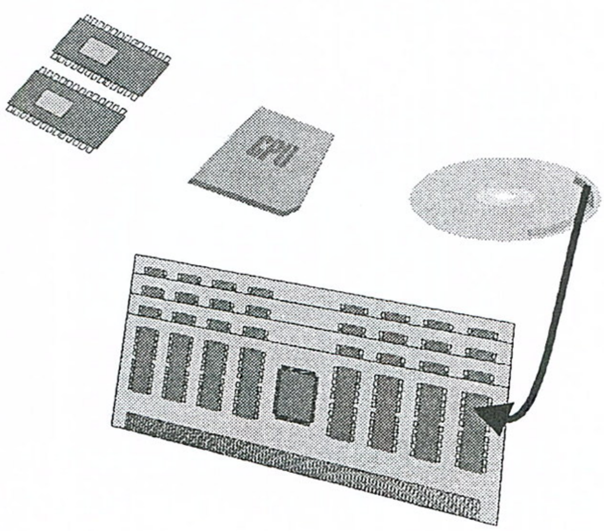


- A DOS alatt a SYSINIT a rendszerbetöltő lemez gyökérkönyvtárában keres egy **CONFIG.SYS** nevű fájlt. Ha ez létezik, a SYSINIT megparancsolja az operációs rendszernek, hogy hajtsa végre a **CONFIG.SYS** utasításait. Ezt a fájlt a felhasználó hozza létre. Utasí-

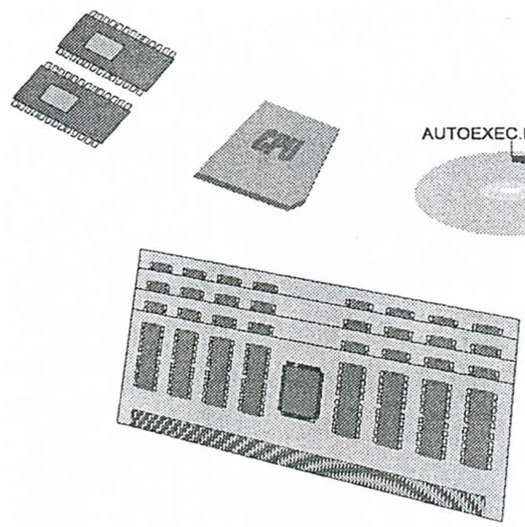
tásai azt közlik az operációs rendszerrel, hogyan kezeljen bizonyos műveleteket, mint például hány fájl lehet nyitva egyszerre. A **CONFIG.SYS** eszközmeghajtók betöltését is kezdeményezheti. Az eszközmeghajtók olyan programokat tartalmaznak, amelyek kiterjesztik a BIOS képességeit a memória és egyes hardvereszközök vezérlésére (*driver*, vagy *memóriarezidens – TSR* programok).



• A SYSINIT közli az **MSDOS.SYS**-szel, hogy töltse be a **COMMAND.COM** fájlt, amely az operációs rendszerhez tartozik, és három részből áll. Egyikük a beviteli/kiviteli funkciók további kiterjesztése. Ez a BIOS-szal együtt kerül a memóriába, és az operációs rendszer részévé válik.



- A **COMMAND.COM** második része a DOS belső parancsait tartalmazza, mint például **dir**, **copy** és **type**. Ez a rész a hagyományos RAM felső végére kerül, ahol felülírhatják az alkalmazási programok, amennyiben az utóbbiaknak memóriahelyre van szükségük. A **COMMAND.COM** más neve *parancsértelmező*.



- A **COMMAND.COM** harmadik része csak egyszer kerül felhasználásra, ezután az operációs rendszer megszabadul tőle. Ez a rész a gyökérkönyvtárban keres egy **AUTOEXEC.BAT** nevű fájlt. Az utóbbi fájlt a felhasználó hozza létre, DOS parancsfájl-utasításokat tartalmaz, és/vagy olyan programok neveit, amelyeket a felhasználó a számítógép minden bekapcsolásakor futtatni akar. A számítógép most már teljesen fel van töltve, és készen áll a használatra.

9. A számítógép használatának első lépése: a DOS

9.1. Bevezetés

Egy számítógép bekapcsolása után az ún. operációs rendszer indul el. Feladata a teljes irányítás. Általában parancsváró üzemmódban vár a felhasználó által kiadott utasításokra, ide tartozik pl. a kívánt programok elindítása, a rendszer egyes beállításainak megváltoztatása, a háttértáron állományok másolása stb. Az operációs rendszer a gépben mindig jelen van és aktív.

A jelenleg elterjedtek közül három jellemzőit nézzük meg közelebbről, ezek a DOS, a WINDOWS és a UNIX.

A legfontosabb különbség a felhasználók és az egyszerre futtatható programok számában mutatkozik meg. Ezek alapján a

- ***DOS egyfeladatos – egyfelhasználós*** *(single user vagy monouser – single tasking),*
- ***WINDOWS egyfelhasználós – többfeladatos*** *(single user – multitasking),*
- ***UNIX többfelhasználós – többfeladatos*** *(multiuser – multitasking)*

rendszer.

A legrégebbi a 60-as években kifejlesztett UNIX. A többfelhasználós – multiuser – rendszer lényege, hogy egy nagyteljesítményű gépre több terminál csatlakozik, melyek előtt ülnek a felhasználók. (A terminál lehet akár ún. buta terminál is, amely egy monitorból és a billentyűzetből áll.) A nagy műveleti sebesség miatt a terminál előtt ülő felhasználó úgy látja, mintha egyedül ő használná a számítógépet (valójában például időosztásos – time sharing – módon csak a gépidő egy részét kapja meg). A UNIX többfeladatos – multitasking -, azaz a felhasználó több programot, taszkot futtathat egy időben. Feladata a felhasználók és taszkok egymástól való védelme, lefagyott taszk rendszerszintű bezárása a többi sérülése nélkül stb. Jelenleg többféle UNIX van forgalomban, ezek egy része (pl. LINUX) szabadon terjeszthető, magas szintű – C nyelven írt – forráskód alakban. UNIX-ot nagygépes környezetben alkalmaznak, az Internet is nagyteljesítményű UNIX számítógépek hálózatából áll.

Az 1981-ben piacra dobott IBM PC személyi számítógépek operációs rendszere az amerikai Microsoft cég nevéhez (is) fűződik. A termék a **DOS (*Disk Operating System – lemezoperációs rendszer*)** nevet kapta. Az assembly nyelven írt DOS a CP/M és a UNIX elemeit tartalmazza. A személyi számítógépek világának megfelelően egy felhasználót tud kiszolgálni, aki egyidőben csak egy programot futtathat. Verziószáma az 1.00-val kezdődött, jelenleg a 7-esnél tart. Léteznek más cégek által kifejlesztett termékek, amelyek a Microsoft MS-DOS rendszerével kompatíbilisek (esetleg jobbak, gyorsabbak), pl. DRDOS, 4DOS, PTSDOS stb. A DOS jelentősége a különféle grafikus felületek (pl. Windows) megjelenésével lecsökkent ugyan, de még nem tűnt el a számítástechnika süllyesztőjében: a Windows 95 tartalmazza, az NT szimulálja.

Az eddig bemutatott UNIX és DOS közös jellemzője, hogy a kívánt parancsot az ún. prompt után betűnként kell beütni, végén egy Enterrel. Ha a parancs nevét elfelejtettük, elírtuk, nem tudjuk pontosan stb., akkor sok minden történhet, csak az nem, amit szeretnénk. A teljes kommunikáció szöveges képernyőn folyik, mi parancsokat adunk a gépnek, a gép meg vagy teljesíti, vagy visszaüzen. Joggal merült fel az igény, hogy az ember-számítógép találkozási felületét, a képernyőt kényelmesebbé, könnyebben kezelhetővé, felhasználóbarátabbá tegyék: már a 70-es évek közepén kidolgozták a *grafikus felhasználói felületek* alapvető elvét. Az első ilyen rendszerek a XEROX, az Apple Lisa (1983) és a Digital Research GEM (1984) volt, ezt követte a Macintosh (1984), majd az IBM OS/2 és a UNIX (X-Window). A PC-kre szintén a Microsoft fejlesztette ki a Windows-t. A Windows (angolul ablakok – a programok a képernyő meghatározott, téglalap alakú részében – ablakban – futhatnak) egy felhasználóbarát grafikus felület, amelyet assembly és C nyelven írtak. Legelső verziója az 1.0-s volt, 1998 nyarán jelent meg legújabb változata, a Windows 98. Másik ág a Windows New Technology (NT), mely megbízhatóságáról híres. A Windows 98 és NT egyfelhasználós-többfeladatos, hálózatban üzemeltetve többfelhasználós-többfeladatos rendszer.

Az eddig ismertetett témakörök közül a DOS és DOS-os segédprogramok gyakorlására szolgálnak a „Bevezetés a számítástechnikába” című tantárgy gyakorlatai. A Windows 98, a UNIX, az Assembly és a C nyelvet későbbi tantárgyaink alkalmával ismerhetik meg közelebbről.

9.2. MS-DOS gyakorló feladatok

A Windows (3.1, 95 és 98) használatakor – bármennyire is el szeretnénk kerülni – óhatatlanul szembetalálkozunk a DOS-szal (könyvtár- és fájlszerkezet, pl. mentéskor, új program telepítésekor stb.). Amennyiben nem ismerjük az alapelveket, nem fogjuk tudni megfelelően használni a Windowst és kellemetlen percekben, számos hibaüzenetben lehet részünk. Emiatt a Windows birtokbavétele előtt feltétlenül javasoljuk a DOS alapjainak megismerését.

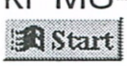
A DOS interaktív környezete a képernyőn megjelenő ún. **promptból** (pl. "`C:\>`") és a **villogó kurzorból** áll. A prompt jelzi azt, hogy a gép parancsra vár, a kurzor pedig azt, hogy a leütött billentyűink

a képernyőn hol jelennek meg. A parancsbeírás módja: betűnként beütjük, amit akarunk, és a véglegesítéshez, a tényleges bevitelhez üssük le az Enter billentyűt. *Mit lehet a prompt-ba beírni? Kiadhatunk fájlokra, könyvtárakra stb. vonatkozó parancsot, vagy valamilyen programot indíthatunk el a nevének beírásával, mást nem tudunk csinálni.* Az **Enter** kiadása után a parancsértelmező megvizsgálja, hogy amit beírtunk, parancs-e. Ha igen, végrehajtja. Amennyiben egyik ismert paranccsal sem egyezik meg, megvizsgálja, létezik-e ilyen néven futtatható program. Ha igen, elindítja, azaz a program megkapja a vezérlést, és kilépés után újból megjelenik a prompt. Ellenkező esetben az egyik leggyakrabban megjelenő hibaüzenetet kapjuk: "**Rossz parancs vagy fájlnév.**" ("**Bad command or file name**"), azaz a gép nem tudja eldönteni, hogy parancsot akartunk kiadni, vagy programot indítani! Egyébként e hibaüzenet megjelenésének leggyakoribb oka az elgépelés.

A most következő feladatsor az MS-DOS parancsainak jelentős részét gyakoroltatja be. E feladatok elvégzése után *közepes mélységben* képet kap a DOS kezeléséről. A példák között néhány olyan finomságot is találhat, amelyet kevesebben ismernek.

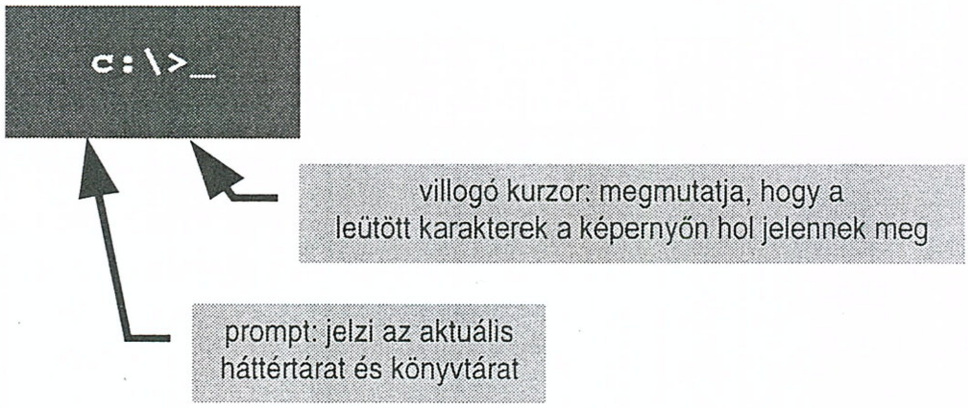
Kezdjük is el a gyakorlást! A feladatok után zárójelben megtalálható a megoldás, így otthon, önállóan is gyakorolhat. Az ""-k között található szöveget a billentyűzeten keresztül kell bevinni a gépbe, tehát az idézőjeleket *nem* kell begépelni! **DOS-parancsoknál a kis- és nagybetűket nem különbözteti meg a gép, egy parancsot akár vegyes betűnagysággal is ki lehet adni.** Ebben a feladatsorban az összes parancsot kisbetűvel írtuk le. A feladatok egymásra épülnek, sorrendben haladjon rajtuk végig!

9.3. DOS I. (alapok)

- Kapcsolja be a számítógépet! Ha a gépen csak DOS van, nem kell semmit sem tennie, rövid idő múltán megjelenik a prompt. Windows használata esetén várja meg, amíg felépül a rendszer és lépjen ki MS-DOS promptba: Windows 95/98 esetén egérrel kattintson a 

gombra, válassza a *Programok* sort, ezen belül az *MS-DOS Parancssor*-t. Elképzelhető, hogy a számítógépén nem található e két sor. Ekkor a *Start* gombra megjelenő menüből válassza a *Futtatás...*-t, írja be: **dosprmpt** és üssön Entert. Ha egy kis ablakban jelenik

meg a "fekete" képernyő, nyomja le a *bal* Alt-ot és üssön Entert. Így átváltunk teljes képernyős megjelenítésre. A promptba addig írjuk be a `cd..` (és Enter) sorokat, amíg a gyakorló feladatsor kezdőállapotát meg nem kapjuk:



- Mielőtt elkezdenénk a gyakorlást, próbálja ki a billentyűzetet! Egyszerre két aktív ún. *kódlap* van a számítógépen: egy angol és egy magyar. Tehát egy-egy billentyű nem csak azokat a betűket jelenti, amelyek rá vannak festve. Az angol karakterkészletet a Ctrl+Alt+F1 billentyűhármas megnyomásával választhatjuk ki, a magyart Ctrl+Alt+F2-vel. Magyar kódlap használata esetén az ékezetes karakterek az írógép elrendezéshez hasonlóan a billentyűzet jobb oldali részén találhatók (kivéve az í-t).

Gyakorolja a kódlapváltást! A feladatmegoldás során mindkettőre szükség lesz (de ha nincs magyar, nem baj). A Ctrl+Alt+F1 használata: nyomja le a Ctrl és az Alt gombokat, egyszer *határozottan* üsse le az F1-et, majd engedje fel a két váltóbillentyűt. Más kódlapkezelő segédprogram használatakor más billentyűkombinációval lehet váltani. Pl. ha angol Windowst használunk és az elterjedt magyar fejlesztésű *Multikey* van a gépen, a váltás Ctrl+F1 és Ctrl+F2-vel történik.

A régi írógépekkel ellentétben ***a nulla és az O betű két külön billentyű, nem helyettesíthetik egymást!*** *A képernyőn a nulla általában át is van húzva:* 0. *Különbözik egymástól az 1 és az L betű is.*

- Első feladatként írassuk ki a DOS verziószámát!

(Megoldás: írjuk be a "`ver`" parancsot, azaz üssük le a V, E és R betűket, majd a bevitel befejezéseként Entert. Ha gépelés közben melléütünk, a Backspace törli a legutoljára bevitt karaktert.)

```
C:\>ver

Windows 98 [Verziószám: 4.10.1998]

C:\>_
```

A gép kiírta a futó operációs rendszer verziószámát. Erről annyit érdemes tudni, hogy 1981-ben az 1.00-s DOS-szal kezdődött. Jelenleg az angol 6.22-es, vagy az angol/magyar Windows 95 (98) szöveggel bejelentkező változatok a legelterjedtebbek (7-es DOS).

- A számítógépben van egy akkumulátorral táplált belső óra, amely kikapcsolt állapotban is működik. Tárolja a mindenkori dátumot és az időt. Nézzük meg, milyen dátumot mutat!

("**date**" parancs és Enter. Látható, hogy a gépben van egy ”örök-naptár” is, kiírja a nap nevét is. Nézzük meg, hogy 1999/2000 fordulója milyen napra esik! Írjuk be: **99.12.31**, Enter, majd ismét **date** és Enter. Láthatjuk, hogy péntekről szombatra virradó éjszaka. Állítsuk vissza a dátumot a mai napra, végül a **date** parancs harmadszori kiadásával ellenőrizzük a visszaállítást – a végén üssünk egy Entert.)

```
A napi dátum: Szo 1998.07.11
Adja meg az új dátumot (éé.hh.nn):99.12.31

C:\>date
A napi dátum: P 1999.12.31
Adja meg az új dátumot (éé.hh.nn):98.07.11

C:\>date
A napi dátum: Szo 1998.07.11
Adja meg az új dátumot (éé.hh.nn):

C:\>_
```

Angol nyelvű operációs rendszer esetén más szöveg jelenik meg és a dátumot más sorrendben kell megadni!

(Ötlet: nézzük meg, milyen napon születtünk! Vigyázat: ha az évszám 80-nál kisebb, vagy hibaüzenetet kapunk, vagy – Windows 98-nál – a kétezerrel kezdődő évszázadra veszi a gép!)

- Most nézzük meg a pontos időt is! A belső óra sohasem pontos: másodpercre pontosan állítsuk be saját óránk alapján!

("**time**" parancs, Enter, írjunk be egy néhány másodperccel későbbi pontos időt, pl. **12.02.05**, majd amikor óránk eléri ezt az időt, üssünk Entert. Így a belső óra a pontos időt fogja mutatni.)

```
C:\>time
Pillanatnyi idő: 12.01.42,91
Adja meg az új időt:12.02.05

C:\>time
Pillanatnyi idő: 12.02.06,76
Adja meg az új időt:

C:\>_
```

Az eddig bemutatott órával kapcsolatos parancsokat (date, time) ritkábban használjuk. Érdemes azonban havonta kiadni, mivel ezek a belső órák általában nem túl pontosak. Előfordult már olyan eset, hogy a nagymértékű késés miatt még az előző napot mutatta a rendszer, s ez gondot okozott egy könyvelő programban.

Más, fontos téma következik: a könyvtárszerkezet, fájlműveletek. Lényege: a háttértáron tárolt "információcsomagok", **a fájlok** nem ömlesztve helyezkednek el, hanem bizonyos tulajdonságok alapján csoportokban – **könyvtárakban tárolhatók** (a fájl meghatározására rövidesen rátérünk). Így az egy-egy feladathoz tartozó fájlok más-más könyvtárakban helyezkednek el, áttekinthetően. A könyvtárszerkezetet különféle utasításokkal saját ízlésünknek megfelelően alakíthatjuk ki. Erre nézzünk példát:

- Hozzuk létre a következő könyvtárszerkezetet és lépjünk be a **GYAK** alkönyvtárba:

```
      C:\
       |
     GYAK
     /   \
    /     \
 ADAM     EVA
```

("**md gyak**", "**cd gyak**", "**md adam**", végül "**md eva**" – Az md a make directory – könyvtár létrehozása, a cd a change directory – könyvtárváltás szavak rövidítése. A parancsok után meg kell adni az (al)könyvtár nevét is. Mivel az **ADAM** és **EVA** a **GYAK**-ból nyílnak, ezért kellett a **GYAK** alkönyvtárba belépnünk. Vigyázzunk: magyar kódlapnál az Y és Z fel van cserélve!)

A képernyő most így néz ki:

```
C:\>date
A napi dátum: P 1999.12.31
Adja meg az új dátumot (éé.hh.nn):98.07.11

C:\>date
A napi dátum: Szo 1998.07.11
Adja meg az új dátumot (éé.hh.nn):

C:\>time
Pillanatnyi idő: 12.01.42,91
Adja meg az új időt:12.02.05

C:\>time
Pillanatnyi idő: 12.02.06,76
Adja meg az új időt:

C:\>md gyak

C:\>cd gyak

C:\gyak>md adam

C:\gyak>md eva

C:\gyak>_
```

legalsó, 25. sor

Általában 25 sorból és soronként 80 karakterből áll, a legalsó sor elérése esetén a sorok felfelé mozognak, ez az ún. *scrollozás* – a scroll angolul (papír)tekercs.

Látható, hogy a prompt megváltozott "`C:\GYAK>`"-ra. **A prompt mindig jelzi az aktuális meghajtót és azt az alkönyvtárat,** amelyben tartózkodunk. Az elején a "**C:**" a merevlemezre (winchesterre) utal. Ha hajlékonylemezre (floppyra) dolgozunk, "**A:**", vagy "**B:**" szerepel, CD-ROM esetében általában D, hálózatnál F-től Z-ig bármi előfordulhat. A mostani gyakorlat során végig a merevlemezre dolgozunk, a prompt mindig "**C:**"-vel fog kezdődni.

- A képernyőre elég sok üzenet került kiírásra, ez néha zavaró lehet. Lehetőségünk van a képernyőt letörölni. Töröljük le!

("`cls`" – jelentése **CLear Screen,** azaz a képernyő törlése)

- Az EDIT szövegszerkesztő segítségével hozzon létre "`EGY.TXT`" néven egy szövegfájlt, melynek tartalma a következő sor: "Ha kivágtad az utolsó fát,".

(Ehhez egy programot fogunk most elindítani. Írjuk be: "**`edit egy.txt`**" végén Enter. Az operációs rendszer háttérbe vonul, eltűnik a prompt és a vezérlést teljes mértékben megkapja az elindított szövegszerkesztő program. Gépeljük be: "**`Ha kivágtad az utolsó fát,`**" – lehetőleg ékezetes betűkkel -, a végén *ne üssünk* Entert. Nyomjuk meg az Alt+F-et (a szürke Alt gombot kell megnyomni, röviden leütni az F betűt, utána felengedni az Alt-ot): beléptünk a Fájl menübe. A fel/le nyilak segítségével válasszuk ki a Mentés (Save) sort és üssünk Entert. A gép kimenti a szöveget. A kilépéshez ismét Alt+F és válasszuk a Kilépés (Exit) sort. A program véget ér, a vezérlést visszakapja a DOS, megjelenik a prompt.

```
C:\gyak>edit egy.txt

C:\gyak>_
```

A szövegszerkesztő indításakor új (kék) képernyőt kapunk. Kilépéskor visszakapjuk az eredeti képernyőtartalmat.

A létrejött fájlunk neve röviden **`EGY.TXT`**, teljes névvel (ún. elérési úttal) pedig **`C:\GYAK\EGY.TXT`**. A teljes név megadásánál a gyökérkönyvtárból indulva el kell jutni a fájl alkönyvtáráig és közben fel kell sorolni az összes könyvtárat, amelyen ezalatt "áthaladunk".

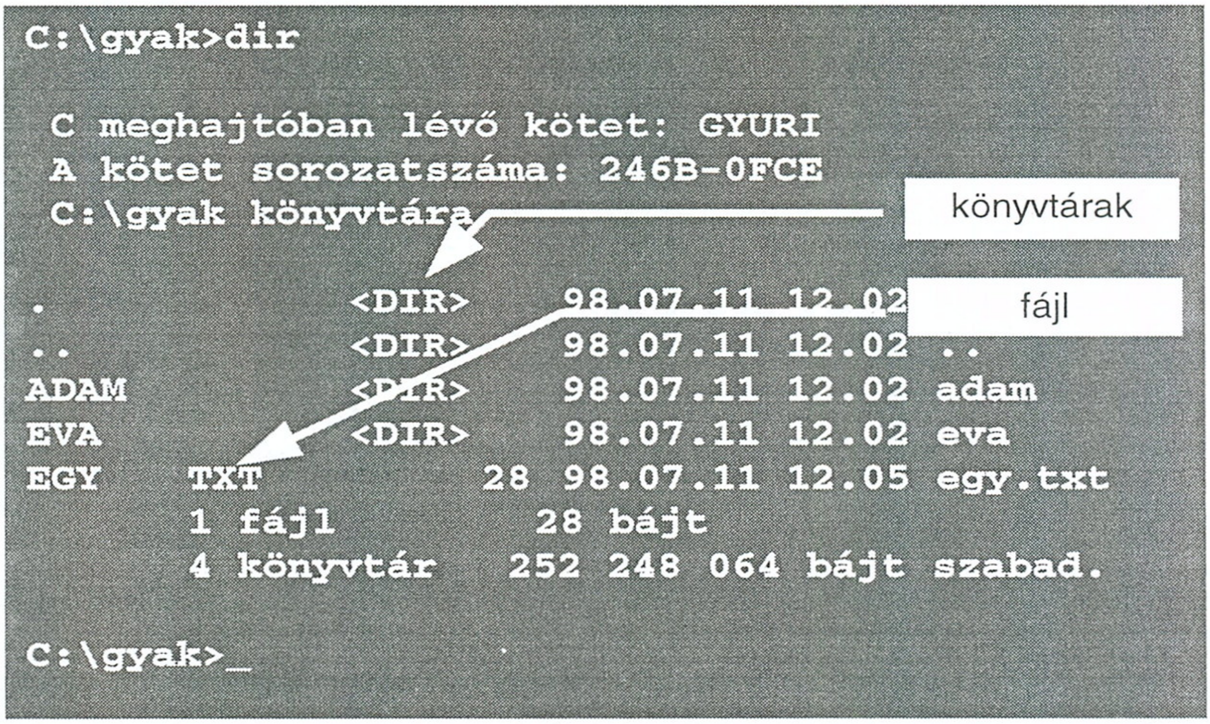
A ***fájl***, vagy állomány (*file*) meghatározása röviden: **bizonyos szempontból összetartozó információk halmaza.** Amit az imént írtunk, logikai szempontból összetartozik, így egy fájlba került. A különféle programok, adataink stb. mind-mind külön fájlok. Minden **fájlnak van egy neve,** amellyel hivatkozhatunk rá. DOS rendszerben a név **maximum 8 karakterből állhat,** amelyet legfeljebb 3 betűs kiterjesztés követhet (ponttal elválasztva, pl. **`EGY.TXT`**, kiejtve "egy pont téikszté"). A kiterjesztés tetszőleges lehet, azonban általában utal a fájl típusára, ezért választottuk most a **`.TXT`**-t, amely a text – egyszerű szövegfájl rövidítése. A fájl neve alfanumerikus karaktereket – betűket és számokat – tartalmazhat, valamint néhány egyéb jelet, pl. "-", "!". Későbbi bosszankodás elkerülése végett általános szabályként ***ne használjunk ékezetes karaktereket!*** Így a feladatsor további részében létrehozandó fájloknál is **`KETTŐ.TXT`**, **`HÁROM.TXT`** stb. helyett **`KETTO.TXT`**, **`HAROM.TXT`** neveket használjunk!) Egy könyvtárban nem lehet két azonos nevű és kiterjesztésű fájl. A fájl a háttértáron elhelyezett információk egysége, a DOS utasításaival *nem lehet* bizonyos részeihez hozzáférni, csak a teljes állományt lehet másolni, mozgatni, többszörözni, törölni, memóriába betölteni (természetesen a különféle programnyelvek fájlkezelő utasításai megengedik a bájtonkénti hozzáférést is). A könyvtáraknak is lehet kiterjesztése, bár ezt általában nem használják ki.

Emlékeztetőül: a ROM-ban, RAM-ban és a különféle háttértárakon tárolt információk alapegysége a bájt (byte). Egy bájt egy 0 és 255 közé eső egész szám. Ez a furcsa érték úgy jön ki, hogy a bájt 8 bitből épül fel. Egy bit lehet 0, vagy 1, ezt a két állapotot elektronikusan könnyű megvalósítani (két pont között ha van feszültség, logikai 1, ha nincs, 0). 1 KB=1024 bájt, 1 MB=1024 KB ("kilobájt", "megabájt"). Az előbb elkészített szövegünkben minden betű egy byte helyet foglal, összesen 26 bájt, az automatikusan berakott soremelés és fájlvége jel miatt e állomány hossza 28 bájt lesz, ahogy mindjárt látni is fogjuk.)

- Az egyik leggyakrabban használt parancs következik, egy könyvtár tartalmának megjelenítése. Listázza ki az alkönyvtár tartalmát a **`dir`** – directory, katalógus paranccsal!

 ("**`dir`**")

A képernyőn a következőt láthatjuk:



A DOS jellegzetessége, hogy a dir parancsra csak egy szintet láthatunk, egy-egy alkönyvtár tartalmát. Látható az ebben az alkönyvtárban található **fájlok neve és kiterjesztése** (itt csak az `EGY.TXT` van), hossza bájtban (28 bájt), valamint a **létrehozás dátuma és ideje** a belső óra adatai alapján. Ugyancsak megjelennek az ebből az alkönyvtárból nyíló újabb alkönyvtárak nevei (itt az `ADAM` és `EVA`), valamint a legelején egy "." és egy ".." kezdetű sorok. Ezek a DOS belső felépítéséből adódó speciális bejegyzések, a gyökérkönyvtár kivételével minden könyvtárban szerepelnek, használatukra később nézünk példát. Végül látható, hogy összesen hány "egység" található az alkönyvtárban (egy fájl és a speciális bejegyzésekkel együtt négy könyvtár), ezek összes mérete (most 28 bájt), végül a háttértár szabad területe bájtban (kb. 252 MB).

- Többféle módon tudunk listázni. Próbáljunk ki egy másik lehetőséget is!

(**"`dir/w`" – a `dir`** parancsot a w *paraméter*rel adtuk ki, az állományoknak csak a neve fog megjelenni. Az osztásjel megtalálható a jobb oldali numerikus billentyűzetrészen, közvetlenül a LED-ek alatt.)

- **Ha listázás közben az adatok eleje a képernyőről kifutna, a** "`dir/p`" oldalakra töri (most nem fut ki, úgyhogy ha kipróbáljuk, a `dir`-rel megegyező végeredményt kapunk).

- Most írjunk be a promptba valami értelmetlenséget, pl. "`fgezuig hjfif hjefio`". Az Enter kiadása előtt rájövünk, hogy semmi értelmeset nem gépeltünk be, a gép a "`Rossz parancs vagy fájlnév.`" ("`Bad command or file name`") hibaüzenetet adná ki. A Backspace-szel letörölhetnénk az egészet, de az ehhez szükséges néhány másodperc túl hosszú idő. A megoldás: üssük le a bal felső sarokban található, különálló szürke "Esc" billentyűt (escape=menekülés)! Mindaz amit eddig írtunk, semmis. Innen folytassuk:

- A biztonság kedvéért nézzük meg az elkészült állomány tartalmát! A szükséges parancs a `type`!

("`type egy.txt`" – **a `type` parancs szöveges állományok tartalmát jeleníti meg a képernyőn. Paraméternek a megtekinteni kívánt fájl nevét kell megadni.** Most pont abban az alkönyvtárban állunk, ahol az állományunk is elhelyezkedik, úgyhogy nem szükséges az elérési út megadása.)

```
C:\gyak>type egy.txt
Ha kivágtad az utolsó fát,

C:\gyak>_
```

(Ha nem a fenti eredményt kaptuk – pl. hibaüzenetet ír ki a gép, vagy nem a `GYAK` alkönyvtárban vagyunk – valamit elrontottunk. Kezdjük *elölről* az egészet és figyeljünk jobban oda.)

- Lépjünk át az `ADAM` alkönyvtárba!

("`cd adam`" – ezt az utasítást már használtuk egyszer, könyvtár váltására szolgál. Mivel a `GYAK`-ból nyílik az `ADAM`, elég a nevet megadni és máris belépünk. Amennyiben a kívánt könyvtár nem innen nyílik, vagy "feljebb" található, a `cd` után bonyolultabb utasításokat kell kiadni, majd láthatunk erre is példát.)

A prompt megváltozik, "`C:\GYAK\ADAM>`" lesz. A könyvtárstruktúrában egy második szinten elhelyezkedő könyvtárba léptünk át.

- Szövegfájl létrehozására nem csak ez az egy lehetőség – külső program használata – van. A DOS belső szövegszerkesztési lehetőségével hozzunk létre "`KETTO.TXT`" néven egy másik szövegfájlt, melynek tartalma: "Ha megmérgezted az utolsó folyót,".

("**`copy con ketto.txt`**" parancs, ezután írjuk be: "**`Ha megmérgezted az utolsó folyót,`**", a végén az új sor miatt üssünk Entert, a befejezéshez Ctrl+Z-t és Entert, vagy rövidebben F6-ot és Entert. **A `copy` parancs eredetileg fájlok másolására** (többszörözésére) szolgál. Paraméternek meg kell adni a létező fájl és az új fájl vagy könyvtár nevét. Itt névként egy különleges, fenntartott szót, a `CON`-t adtuk meg. Jelentése konzol=billentyűzet. Innen "másolunk" valamit egy fájlba. Így a **`copy` parancs** ezen speciális használatával új szövegfájlokat hozhatunk létre! Előnye, hogy a **`copy` ún. *belső DOS parancs***, azaz a `COMMAND.COM` parancsértelmező hajtja végre, szemben az `EDIT`-tel, mely *külső parancs*, a DOS alkönyvtárban található `EDIT.COM` futtatható fájlt, programot jelenti. A "**`copy con fájlnév`**" segítségével sokszor hamarabb hozhatunk létre egyszerűbb *új* szövegállományokat, mint az `EDIT`-tel, amely komolyabb program, kész fájlokat módosíthatunk vele stb.)

```
C:\gyak\adam>copy con ketto.txt
Ha megmérgezted az utolsó folyót,
^Z
        1 fájl másolása megtörtént.

C:\gyak\adam>_
```

- Végül ugyanígy hozzuk létre "`HAROM.TXT`" néven egy fájlt, melynek tartalma: "Rá fogsz döbbenni, hogy a pénz nem ehető.".

(Üssük le az F3-at, ekkor a promptba beíródik a legutoljára kiadott utasítás! A Backspace-szel töröljük le a "`ketto.txt`"-t és írja be helyette a "`harom.txt`"-t, így a kiadandó parancs: "`copy con harom.txt`" lesz. Enter, "Rá fogsz döbbenni, hogy a pénz nem ehető." szöveg beírása – a végén nem kell Enter -, a kimentéshez F6 és Enter.)

```
C:\gyak\adam>copy con harom.txt
Rá fogsz döbbenni, hogy a pénz nem ehető.^Z
     1 fájl másolása megtörtént.

C:\gyak\adam>_
```

- Listázzuk ki az alkönyvtár tartalmát!

("`dir`" – megjelenik az alkönyvtár tartalma, a 35 bájtos `KETTO.TXT` és a 41 bájtos `HAROM.TXT` szövegfájlok. Ha a fájlok mérete más, nem baj, több betűt vagy Entert írtunk be a kelleténél.)

A már említett "." és ".." bejegyzések használatára a következő feladat mutat példát:

- Listázzuk ki az `ADAM`-ból a `GYAK` alkönyvtár tartalmát kétféle módon, abszolút és relatív úton!

(Két megoldás kínálkozik. Az abszolút utas: "`dir c:\gyak`", a relatív változat: "`dir..`". Próbáljuk ki mindkettőt! A két oldallal korábbi képet láthatjuk ismét kétszer. Magyarázat:)

A "." az aktuális, a ".." az egy szinttel feljebb levő – ún. "szülő" könyvtárat jelenti (parent directory). Bármely utasításban használhatjuk, ahol elérési utat, vagy teljes fájlnevet kell megadni. Van még egy jel: a "\" – backslash -, amely a legfelső szinten levő gyökérkönyvtárat jelenti, pl. a "`dir c:\`"-t rövidebben a "`dir\`"-el helyettesíthetjük. A "\"-t ne keverjük össze a "/" jellel!

A `cd\`, `cd..` mellett használható a `cd...`, `cd....` stb. is, amelyek még több könyvtárat lépnek vissza.

- Fűzzük össze a `KETTO.TXT` és `HAROM.TXT` fájlokat ebben a sorrendben, az új állomány neve legyen `NEGY.TXT`!

("`copy ketto.txt+harom.txt negy.txt`" – a `copy` parancs fájlokat is össze tud fűzni.)

- Nézzük meg a végeredményt!

("`type negy.txt`" – az Entert azért kellett a szöveg végére beírni, hogy új sorba kerüljön az összefűzött szöveg.)

- Listázzuk ki a könyvtár tartalmát!

("`dir`" – a 35 és 41 bájtos fájlokból egy 77 bájtos állomány lett, egy byte-tal több, mint e két szám összege.)

A képernyőn a következőt láthatjuk:

```
C:\gyak\adam>copy ketto.txt+harom.txt negy.txt
ketto.txt
harom.txt
        1 fájl másolása megtörtént.

C:\gyak\adam>type negy.txt
Ha megmérgezted az utolsó folyót,
Rá fogsz döbbenni, hogy a pénz nem ehető.
C:\gyak\adam>dir

 C meghajtóban lévő kötet: GYURI
 A kötet sorozatszáma: 246B-0FCE
 C:\gyak\adam könyvtára

.            <DIR>        98.07.11 12.02 .
..           <DIR>        98.07.11 12.02 ..
KETTO    TXT           35 98.07.11 12.13 ketto.txt
HAROM    TXT           41 98.07.11 13.18 harom.txt
NEGY     TXT           77 98.07.11 13.19 negy.txt
        3 fájl            153 bájt
        2 könyvtár    252 084 224 bájt szabad.

C:\gyak\adam>_
```

- Mozgassuk át a **NEGY.TXT**-t az **EVA** alkönyvtárba! A szükséges parancs a **move** – mozgat!

(Itt is két megoldás kínálkozik. A hosszabb az abszolút út: "**move negy.txt c:\gyak\eva**". A rövidebb a relatív: "**move negy.txt ..\eva**". Írjuk be az egyiket! **A move áthelyezi a fájlt és az eredeti helyről letörli.** Így megfelel egy **copy** és utána egy **del** fájltörlő utasításnak. Igazából nem fizikai másolást végez – mint a **copy** -, hanem a háttértár fájlnyilvántartó táblázatában írja csak át a fájl megtalálási helyét, (így gyorsabb.)

```
C:\gyak\adam>move negy.txt ..\eva
C:\GYAK\ADAM\negy.txt => C:\GYAK\eva\negy.txt [ok]

C:\gyak\adam>_
```

- Listázzuk ki a könyvtár tartalmát!

("**dir**" – a **NEGY.TXT** fájlt áthelyeztük, ebben a könyvtárban már nem található, csak a már felesleges **KETTO.TXT** és **HAROM.TXT** fájlok.)

- Nevezzük át a **KETTO.TXT**-t **TWO.TXT**-re! A szükséges parancs a **ren** (rename – átnevezés)!

("**ren ketto.txt two.txt**" – **a ren-nel fájlokat nevezhetünk át.** Két paramétert igényel: a régi és az új fájlnevet. Könyvtár átnevezésére a **move** parancs szolgál.)

- Nézzük meg a végeredményt!

("**dir**" – a fájl neve megváltozott, **HAROM.TXT** és **TWO.TXT** jelenik meg.)

- Töröljük le egy utasítással a két szövegfájlt! A szükséges parancs a **del** – delete, törlés!

(Két utasítással egyszerű lenne: "del two.txt" és "del harom.txt". Egy utasítással pedig: "**del *.***".)

```
C:\gyak\adam>del *.*
A könyvtárban lévő összes fájl törlődik!
Biztosan törölni kíván (I/N)?i

C:\gyak\adam>_
```

A * és a ? az ún. joker karakterek, segítségükkel több fájlra is vonatkozhat a parancs, így a "*.*" az összes fájlra, a "*.TXT" az összes .TXT kiterjesztésű fájlra vonatkozik stb. A ? egy karaktert helyettesít, pl. a "**?alom**" jelenti mind a malom, mind a halom fájlokat, vagy könyvtárakat. **A törléshez a DOS megerősítést kér,** üssük le az I-t és Entert! Angol DOS esetén I helyett Y-t – yes – kell ütnünk.)

- Így az `ADAM` könyvtárunk üres lett. Szüntessük meg magát a könyvtárat!

(Könyvtár törlésének két feltétele van: üres legyen és ne tartózkodjunk benne. Most töröltünk le mindent, viszont benne vagyunk, tehát ki kell lépni belőle: "`cd..`", ezután adhatjuk ki az "`rd adam`" – remove directory, könyvtár törlése parancsot.)

```
C:\gyak\adam>cd..
C:\gyak>rd adam
C:\gyak>_
```

- Mozgassuk át az itt található `EGY.TXT` fájlt is az `EVA`-ba!

(A hosszabb megoldás: "`move egy.txt c:\gyak\eva`", a rövidebb: "`move egy.txt .\eva`". Figyelem: **a "\" visszafelé döntött osztásjelet – backslash – szükséges használni,** amely a billentyűzeten általában az Enter felett, vagy alatt található, Shift nélkül kell megnyomni. Ha a billentyűzetre magyar betűk vannak festve, a backslash az Ű-n található – természetesen Ctrl+Alt+F1 megnyomása után.)

```
C:\gyak>move egy.txt .\eva
C:\GYAK\egy.txt => C:\GYAK\eva\egy.txt [ok]
C:\gyak>_
```

- Lépjünk át az `EVA` alkönyvtárba!

("`cd eva`" – a prompt `C:\GYAK\EVA`-ra változik.)

- Listázzuk ki az alkönyvtár tartalmát!

("`dir`" – látható az előbbiekben ide mozgatott két szövegfájlunk, a 28 bájtos `EGY.TXT` és a 77 bájtos `NEGY.TXT`.)

- Most az alkönyvtár tartalmát ne a képernyőre, hanem egy `LISTA.TXT` fájlba írassuk ki!

("`dir>lista.txt`". Magyarázat: ez az ún. **átirányítás. Egy** "`>`" **jel használatakor új fájl keletkezik,** "`>>`" esetén a létező végére íródik a tartalom. Itt tehát a katalógus nem a képernyőre íródik ki, hanem egy fájlba.)

• Nézzük meg a fájl tartalmát!
("**`type lista.txt`**" – a fájl tartalmát látjuk, olyan, mintha a **`dir`**-t adtuk volna ki!)

• Töröljük le a fájlt!
("**`del lista.txt`**")

• Az **`EGY.TXT`** végéhez fűzzük hozzá a **`NEGY.TXT`**-t, így végre három részből összerakva egy fájlban megkapjuk a németországi környezetvédők egyik régi jelmondatát, egy ma is aktuális ősi indián mondást:
"Ha kivágtad az utolsó fát,
Ha megmérgezted az utolsó folyót,
Rá fogsz döbbenni, hogy a pénz nem ehető."

("**`type negy.txt>>egy.txt`**" – tehát a "**`>>`**" **a fájl végéhez hozzáfűz valamit**, hasonlóan a **`copy`** + parancshoz.)

Talán kevesen emlékeznek rá, de a magyar ékezetes betűk megjelenítése sokáig nem volt egyszerű (lásd Teletext): vagy semmit, vagy csak néhányat lehetett használni. Az IBM PC-k MS-DOS operációs rendszere teremtette meg az *összes* magyar ékezetes betű használatának lehetőségét.

• Nevezzük át a fájlt **`IDEZET.TXT`**-re és Írassuk ki a tartalmát!
("**`ren egy.txt idezet.txt`**" és "**`type idezet.txt`**")

```
C:\gyak\eva>type negy.txt>>egy.txt

C:\gyak\eva>ren egy.txt idezet.txt

C:\gyak\eva>type idezet.txt
Ha kivágtad az utolsó fát,
Ha megmérgezted az utolsó folyót,
Rá fogsz döbbenni, hogy a pénz nem ehető.
C:\gyak\eva>_
```

Rövid kiegészítő téma következik, a nyomtatás:
Bár manapság leginkább Windows alól történik, természetesen lehetőségünk van DOS-ból is szövegfájljainkat kinyomtatni.

A szükséges parancs a **`print`**, pl.: "**`print fájlnév`**". Más megoldás a "**`copy con`**"-hoz hasonlóan speciális eszköz használata, pl.: "**`copy fájlnév lpt1`**". Ha a képernyőt szeretnénk egy az egyben papíron viszontlátni, a billentyűzet szürke PrintScreen gombját kell megnyomni (egyes gépeken Shift-tel együtt) – Windows alatt mást jelent! Grafikus üzemmód esetén előbb a **`GRAPHICS.COM`** rezidens programot kell elindítani! *A nyomtatás feltételei csupán a következők: a printer megfelelően hozzá legyen kapcsolva a géphez, legyen bekapcsolva, legyen on-line üzemmódban (azaz ha van ilyen nevű gomb, a mellette található LED világítson) és az sem árt, ha papír is van benne.*

- HA VAN NYOMTATÓ a géphez kötve, nyomtassuk ki a szöveget!

("**`print idezet.txt`**" és két Enter. Ha nem megy, a másik módszer: "**`copy idezet.txt lpt1`**")

```
C:\gyak\eva>copy idezet.txt lpt1
        1 fájl másolása megtörtént.

C:\gyak\eva>_
```

- Még egy lehetőség (ha van nyomtatónk, próbáljuk ki): nyomjuk meg a Ctrl+P -t! Ezt követően mindaz amit begépelünk, a nyomtatóra is átküldésre kerül! A kettős üzemmód kikapcsolása egy újabb Ctrl+P -re történik. Egyes nyomtatók ilyenkor még nem adják ki a papírt, csak "gyűjtik az információt". ***A lapdobás kezdeményezése billentyűzetről***: **`copy con lpt1`** (Enter), bal Alt+12 (a 12-t a numerikus billentyűzeten üssük le), F6 és Enter. A con a console – billentyűzet – rövidítése, az lpt1 pedig a printer port neve.

Más téma következik: **egy fájlnak, vagy alkönyvtárnak különböző jellemzői – *attribútum*ai lehetnek. Lehet írásvédett, archív, rendszer, esetleg rejtett jellegű.** A rejtett fájlok, vagy könyvtárak a DOS **`dir`** parancsára nem jelennek meg (csak a "**`dir/ah`**"-ra). Hasznos jellemző az írásvédettség, ekkor ugyanis a **`del`** parancs nem tud törölni. Fontos állományainkat védhetjük ezen a módon a véletlen törlés ellen. Erre nézzünk példát:

- Tegyük az **IDEZET.TXT**-t írásvédetté ("read only" – csak olvasható)! A szükséges parancs az **attrib**!

(Ezt a parancsot nem ismerjük. Milyen paramétereket igényel, és egyáltalán, hol van ez leírva? A megoldás: adjuk ki az "**attrib/?**" parancsot!!! **A "/?" paraméter megadása esetén a gép kiírja a parancs szintaktikáját és a különféle lehetséges paramétereket.** Látható, hogy meg kell adni +/- jellel a jellemző engedélyezését/tiltását, magát a jellemzőt, végül a fájl nevét. Azaz a megoldás: "**attrib +r idezet.txt**".)

```
C:\gyak\eva>attrib/?
Fájlattribútumok megjelenítése vagy módosítása
ATTRIB  [+R|-R]  [+A|-A]  [+S|-S]  [+H|-H]  [[meghaj-
tó:][útvonal]fájlnév] [/S]
   + Attribútum beállítása
   - Attribútum törlése
   R Írásvédett fájl attribútum
   A Archív fájl attribútum
   S Rendszerfájl attribútum
   H Rejtett fájl attribútum
   /S Fájlok feldolgozása a megadott elérési út minden
könyvtárában
C:\gyak\eva>attrib +r idezet.txt

C:\gyak\eva>_
```

- Írassuk ki az alkönyvtár fájljainak attribútumait!

("**attrib**" – az **IDEZET.TXT** "írásvédett", más néven "csak olvasható" – R – lett. A fájlok alapértelmezésben archív – A - attribútumot kapnak, ezt is láthatjuk.)

- Töröljük le egy utasítással az alkönyvtár fájljait!

(A lehetőségek: az összes fájlt törlő "**del *.***", a vele egyenrangú "**del.**", vagy – mivel csak két szövegfájl van a könyvtárban -, a "**del *.txt**". Használja közülük valamelyiket. Ha a DOS megerősítést kér, üssünk I-t és Entert.)

```
C:\gyak\eva>del.
A könyvtárban lévő összes fájl törlődik!
Biztosan törölni kíván (I/N)?i

C:\gyak\eva>_
```

- Listázzuk ki az alkönyvtár tartalmát!

("**`dir`**" – a **`NEGY.TXT`** eltűnt, de *az írásvédett* ***`IDEZET.TXT`-t*** *nem törli le a* ***`del`*** *parancs!*)

A DOS korábbi verzióiban ha egy fájlt letöröltünk, és később megbántuk, csak nehezen. segédprogram segítségével tudtuk esetleg visszanyerni. **Az 5-ös verziótól a letörölt állomány visszaállítása egyszerűbbé vált, megjelent az `undelete` utasítás.** Az alapelv a következő: egy fájlt a DOS *csak logikailag* töröl le: a nyilvántartásban a nevének első karakterét egy 229-es kódú karakterre ("σ") cseréli és a fájl által elfoglalt területet szabadnak nyilvánítja. **Amennyiben nem végeztünk a háttértáron írási műveletet** (vagy defragmentálást – lemeztöredezettség-mentesítést)**, a törölt állomány visszaállítható.** Minél több írási műveletet (fájlmásolás, mozgatás stb.) végzünk, a visszaállítás esélye egyre csökken. A parancs szintaktikája általános esetben: ”**`undelete`**”, ha ismerjük a fájl nevét: ”**`undelete` fájlnév**”.

- Ha ”sima” DOS alatt dolgozunk, adjuk ki az ”**`undelete negy.txt`**” parancsot és állítsuk vissza a törölt fájlt! Windows MS-DOS Parancssor használata esetén – a nehezebb használat miatt – *ne csináljunk semmit*, lépjünk a következő feladatra!

Fontos! A törlés visszaállítása abban az esetben már nem lehetséges, ha pl. a **`copy`** parancs használatával írtunk felül egy létező fájlt! Ekkor az `undelete` nem adható ki, mert a felülírás miatt *létezik* egy fájl azon a néven!

- Másoljuk át az **`IDEZET.TXT`**-t egy könyvtárral feljebb, a **`GYAK`**-ba!

(Mivel most az **`EVA`**-ban vagyunk és a **`GYAK`** a szülőkönyvtár, használhatjuk a relatív utat, a megoldás: "**`copy idezet.txt ..`**" – ez a **`copy`** hagyományos használata, fájl másolása egy másik könyvtárba. A ”cél”fájl írásvédettségét a **`copy`** parancs megszünteti.)

- Lépjünk is át a **`GYAK`** alkönyvtárba! Nem bízunk a másolásban, hasonlítsuk össze az eredetivel!

(Az előző meggondolások alapján: "**`cd..`**", utána "**`fc idezet.txt .\eva\idezet.txt`**".)

- Lépjünk át a gyökérkönyvtárba!

("`cd\`" – ez az utasítás akármilyen mélyen vagyunk, a gyökérkönyvtárba visz.)

```
C:\gyak\eva>copy idezet.txt ..
        1 fájl másolása megtörtént.

C:\gyak\eva>cd..

C:\gyak>fc idezet.txt .\eva\idezet.txt
idezet.txt és .\eva\idezet.txt fájl összehasonlítása
FC: Nincsenek különbségek.

C:\gyak>cd\

C:\>_
```

- A `dir` parancs segítségével keressük meg az `IDEZET.TXT` fájl pontos helyét!

(A sima `dir` erre nem alkalmas. Valamilyen paraméterrel kell kiadni a parancsot, de melyikkel? Adjuk ki itt is a "`dir/?`" parancsot! Látható, hogy az S paraméterre van szükség, ekkor keres az alkönyvtárakban is. Tehát a megoldás: "`dir idezet.txt/s`". A gép meg fogja találni a `C:\GYAK` és a `C:\GYAK\EVA` könyvtárakban található `IDEZET.TXT` fájlokat, tehát mindkét állományt.)

```
C:\>dir idezet.txt/s
  C meghajtóban lévő kötet: GYURI
  A kötet sorozatszáma: 246B-0FCE

C:\gyak könyvtára
IDEZET   TXT        104     98.07.11 13.36 idezet.txt
         1 fájl         104 bájt

C:\gyak\eva könyvtára
IDEZET   TXT        104     98.07.11 13.36 idezet.txt
         1 fájl         104 bájt

Az összes felsorolt fájl:
         2 fájl         208 bájt
         0 könyvtár 242 581 504 bájt szabad.

C:\>_
```

- A `/?` paraméterrel máshol is kaphatunk segítséget. Próbáljuk ki: **`cls/?`** és **`date/?`** !

- Csak a DOS parancsoknál van még egy segítségkérési lehetőség, ez a **`Help`** program használata. Írjuk be a "**`help`**" parancsot! A megjelenő menüből válasszuk ki az **`Attrib`**-ot, olvassuk el a kiírt szöveget. Nézzük meg a "Notes" és az "Examples" menüket is! Részletesebb segítséget kapunk, mint a "/?" használatával. Kilépés: Alt+F és Kilépés (Exit).

Fontos különbség, hogy a "`parancsnév/?`" a futtatható programot indítja el a "/?" paraméterrel. Ha a program készítői megírták, akkor segítő információkat ír ki a képernyőre. Így a nem DOS-hoz tartozó utasításokról (tömörítő programok stb.) is kaphatunk információt, teljesen ismeretlen programot is megpróbálhatunk ezzel a paraméterrel indítani. **Ezzel szemben a "`help`" a DOS tartozéka, külső segédprogram**, amely a DOS parancsait tartalmazza, meghatározott parancsokra nyújt csak választ.

- Ugyanígy keressük meg az **`EVA`** alkönyvtárat is!
("**`dir eva/s`**" – a gép megtalálja és kiírja, hogy a **`GYAK`**-ból nyílik.)

- Továbbra is a gyökérkönyvtárban tartózkodva töröljük le a **`GYAK`** alkönyvtárat az összes fájljával és alkönyvtárával együtt!
("**`deltree gyak`**", a megerősítéshez I és Enter. VESZÉLYES PARANCS! Segítségével fájlokat, illetve teljes könyvtárstruktúrát törölhetünk, írásvédelemmel és minden mással egyetemben.)

A bevezető feladatsor első részének végére értünk. Visszaállítottuk a kiinduló helyzetet, letöröltünk mindent, amit létrehoztunk. Idáig eljutva alapszinten megismerhettük a különféle könyvtár- és fájlműveleteket, parancsokat. Néhány egyéb parancs van még, megértésük nem nehéz, leírásuk számtalan könyvben megtalálható.

A Windows MS-DOS Parancssorból az **`exit`**-re léphetünk ki, ezután kapcsoljuk ki a gépet (ha Windowsban vagyunk, előtte lépjünk ki belőle)!

9.4. DOS II. (finomságok)

A gyakorló feladatsor második részében a DOS mélyebb lehetőségeit ismerhetjük meg: szűrők, a **find**, **sort**, **more** parancsok, batch fájlok készítése stb. E parancsok használatával a harmadéves *Unix operáció rendszer* tantárgy gyakorlatain újból találkozhatnak.

- Kapcsoljuk be a gépet, ha kell lépjünk ki az MS-DOS Parancssorba és lépjünk át a főkönyvtárba (gyökérkönyvtár – *root directory*). A kiindulási helyzet ismét:

```
C:\>_
```

- Hozzuk létre ismét a **GYAK** alkönyvtárat és lépjünk be!

("**deltree gyak**", "**md gyak**" és "**cd gyak**" – először letöröljük az esetlegesen létező könyvtárat, utána hozzuk létre, hogy biztosan üres legyen.)

- Indítsunk el egy memóriarezidens segédprogramot, a **doskey**-t!

("**doskey**" – lehet, hogy már telepítve van és más üzenetet kapunk – nem baj. A válaszüzenet után látszólag nem történik semmi, visszakapjuk a promptot.)

```
C:\gyak>doskey
Telepített DOSKEY.

C:\gyak>_
```

- Hozzunk létre egy **FRUITS** nevű fájlt (ne legyen kiterjesztése) és néhány gyümölcsnevet írjunk be, egymás alá!

("**copy con fruits**", a szavak beírása, a legutolsó után üssünk Entert – egy üres sort hoz létre, ami most kell -, kilépés Ctrl+Z és Enter, rövidebben F6 és Enter. A Ctrl+Z jelzi a fájl végét: *End of File – EOF*.)

```
C:\gyak>copy con fruits
sárgabarack
körte
ananász
kókuszdió
görögdinnye
^Z
  1 fájl másolása megtörtént.

C:\gyak>_
```

Ha egy fájlnak nincs kiterjesztése, ritka a DOS-ban, de megoldható. Ugyanígy a könyvtáraknak *néha* szoktak kiterjesztést adni. Mindenesetre a következő utasítások eleinte furcsának tűnhetnek.

• Listázzuk ki a fájl tartalmát!
("`type fruits`")

A szűrőkről (csővezeték, pipeline). **Jele: "|".** A billentyűzeten általában az Enter körül található gomb, a Shift-tel együtt kell megnyomni. Jelentése a következő: **a promptba több parancsot is beírhatunk, szűrőkkel elválasztva.** A szűrő előtti parancs végeredménye – kimenete – a szűrő utáni parancs bemenetére kerül. A szűrőket DOS-ban korlátozottan használhatjuk, UNIX-ban többszörös, bonyolult szűrőműveleteket is végezhetünk. Ha a billentyűzetre magyar betűk vannak festve, a szűrő az Ű-n található – természetesen Ctrl+Alt+F1 megnyomása után, Shift nyomvatartásával.

• Nézzünk példákat a szűrők használatára! Listázzuk ki a fájl tartalmát *ABC sorrendben*!
("`type fruits|sort`" – **a type így nem a képernyőre ír ki, hanem a fájl tartalma a sort-nak adódik át,** amely egyszerűen növekvő sorrendbe rakja a bemeneti információt – lásd "`sort/?`". A parancsot ne gépeljük be, hanem nyomjuk meg a kurzor fel (↑) billentyűt, így megkapjuk az előzőleg beírt parancsokat – pl. `type fruits` -, és a végére beírjuk még amit szükséges. Erre jó a `doskey`!)

```
C:\gyak>type fruits|sort
ananász
görögdinnye
kókuszdió
körte
sárgabarack

C:\gyak>_
```

- Van még egy megoldás, ezt is próbáljuk ki: "`sort<fruits`" (átirányítás: a bemenetet jelöli a "<" jel, egy programnak így adhatunk meg fájlt bemenetként. A kimenetet nem irányítottuk át, így a *standard output*ra, a képernyőre történik. Az eredmény ugyanaz.)

- Most egy utasítással érjük el, hogy a **`FRUITS`** fájl gyümölcsnevei ABC sorrendben ne a képernyőn rendeződjenek, hanem a **`FRUITS1`** nevű fájlban!

(A **`doskey`** – kurzor fel – segítségével jelenítsük meg az előzőleg beírt parancsot és bővítsük ki. A kiadandó parancs: "`sort<fruits>fruits1`" – a **`sort`** bemenetét átirányítottuk a *standard input*ról (billentyűzet) egy fájlba, a kimenetet is átirányítottuk a *standard output*ról (képernyő) egy másik fájlba. Ebben a megoldásban nem adhatjuk meg ugyanazt a fájlnevet kimenetnek. A prompt rögtön visszatér, a parancs nem ír ki a képernyőre semmit. Nézzük meg az elkészült fájl tartalmát a "`type fruits1`" utasítással!)

```
C:\gyak>sort<fruits>fruits1

C:\gyak>type fruits1
ananász
görögdinnye
kókuszdió
körte
sárgabarack

C:\gyak>_
```

• Itt is van még egy megoldás. Adjuk ki a "**type fruits|sort>fruits**" parancsot (egy szűrő és egy átirányítás)! Az eredeti fájlt átírjuk az ABC sorrendbe rendezettre. Megtekintése: "**type fruits**":

```
C:\gyak>type fruits|sort>fruits

C:\gyak>type fruits
ananász
görögdinnye
kókuszdió
körte
sárgabarack

C:\gyak>_
```

• Írassuk ki a **FRUITS** azon sorait, amelyek tartalmaznak "e" betűt! A szükséges parancs a **find**!
("**find/?**", áttanulmányozás után a megoldás: "**find/n "e" fruits**" – a keresett szöveget idézőjelek közé kell tenni!)

• Most csak azt írassuk ki, hány sor tartalmazza az "e" betűt!
("**find/c "e" fruits**" – a gép kiírja a helyes választ: 2 sorban található "e" betű.)

• Töröljuk le az alkönyvtár tartalmát egy utasítással úgy, hogy a gép ne kérdezzen semmit!
(A "del *.txt" nem jó, mert a **FRUITS** és a **FRUITS1** kiterjesztés nélküli fájlok, nem fogja letörölni. Marad a "del *.*" és a "del." utasítások, ekkor viszont a gép megerősítést kér, amelyre az "y"-t és Entert kell válaszolni. Ezért a szűrő segítségével egy kis trükköt vetünk be, a megoldás: "**echo i|del.**" – **a szűrő átadja az i-t a del-nek, így a gép nem vár billentyűleütésre, azonnal letörli az összes fájlt!!!** Angol DOS esetén i helyett y-t – yes – üssünk.)

```
C:\gyak>find/n "e" fruits

---------- fruits
[2]görögdinnye
[4]körte

C:\gyak>find/c "e" fruits

---------- fruits: 2

C:\gyak>echo i|del.
A könyvtárban lévő összes fájl törlődik!
Biztosan törölni kíván (I/N)?i

C:\gyak>_
```

• A szűrők használatára még egy példa: a `dir/p`-hez hasonló hatású a **`more`** parancs, azzal a különbséggel, hogy bármely utasítás után használhatjuk. Adjuk ki a gyökérkönyvtár megjelenítésére a "`dir\|more`" parancsot!

• Lazításképpen: nem tetszik nekünk a prompt, túl száraz. Változtassuk meg!

("`prompt/?`", utána "`prompt Éljenek a sárga mikiegerek!$_$p$g`")

```
C:\gyak>prompt Éljenek a sárga mikiegerek!$_$p$g

Éljenek a sárga mikiegerek!
C:\gyak>_
```

Mialatt különféle utasításokat adunk a gépnek, a már említett COMMAND.COM parancsértelmező fájllal kommunikálunk. Ez a program hozza létre a **prompt**-ot, fogadja parancsainkat, indítja el a kívánt programokat, egyszóval egy interaktív felületet nyújt. A gépben viszont nem csak ez az egy program fut, hanem az úgynevezett ***memóriarezidens*** programok is (a DOS ettől még egyfeladatos – single tasking -, azaz egyidőben csak egy programmal tud foglalkoz-

ni). **A rezidens programok a háttérben futó különféle segédszoftverek: billentyűzet-, memóriakezelő stb**, részletesen nem foglalkozunk velük.

- Listázzuk ki a rezidens programokat! A szükséges parancs a `mem`!

("**mem/?**", áttanulmányozás után a megoldás: "`mem/c`". Így viszont a képernyőre túl sok sort ír ki a gép, nem láthatjuk az elejét, ezért a még jobb megoldás szűrő alkalmazása: "`mem/c|more`" – **a more parancs oldalanként végzi el a kiírást, a továbblépés Enterre történik.** A memóriarezidens programok között felfedezhetjük a `COMMAND.COM` parancsértelmezőt, a `KEYB` magyar billentyűzetkezelőt, ezenkívül láthatjuk a számítógép memóriájának felosztását, a legnagyobb futtatható program méretét stb.)

- Nézzük meg, a `COMMAND.COM` milyen memóriákat foglal el!

("`mem/?`" után a megoldás: "`mem/m command`" – összesen kb. 7 KB helyet foglal a rezidens része.)

- Indítsuk el a parancsértelmező egy újabb példányát, a "`command`" parancs kiadásával! Ha most a Norton Commanderben dolgoztunk volna, az NC eltűnik. Megjelenik a copyright és a prompt. Adjuk ki: "`mem/c|more`"→ *két* `COMMAND.COM` van a memóriában! Lépjünk ki az "**exit**" paranccsal (a Norton commander újból megjelenik). Végül a promptot állítsuk vissza az eredeti megjelenésre: "`prompt $p$g`" úgy, hogy p-t ütünk, utána F8-at és a megjelenő sort javítjuk át (`doskey` lehetőség).

- A `more` egyébként működése közben a `C:\TEMP` könyvtárban létrehoz ideiglenesen egy fájlt, amit a végén letöröl. Ezt a fájlt elkaphatjuk, ha egy `more`-os parancsot végrehajtás közben a Ctrl+C-vel leállítunk. A fájl jelenlétének bizonyítására adjuk ki sorban a következő parancsokat (a `GYAK` könyvtárban vagyunk):

`dir c:\temp`	(kapunk valamit, jegyezzük meg)	
`dir c:\temp	more>dir.txt`	(fájl létrehozása)
`type dir.txt`	(egy furcsa nevű, 0 bájtos fájl is lesz → ez az!)	
`del dir.txt`	(fájl törlése)	

- Ha egy könyvtárral sokat dolgozunk, egy idő után fárasztó lehet a könyvtár teljes nevét begépelni. Ezen könnyít a **subst**, melynek segítségével egy meghajtó betűjelével hivatkozhatunk könyvtárra.

Adjuk ki a következő parancsokat egymás után:

subst/?	
subst g: c:\temp	(az új meghajtó neve **G:**, a hozzárendelt könyvtár **C:\TEMP**)
dir g:	(a **C:\TEMP** könyvtár tartalmát kapjuk meg!)
subst g: /d	(a látszólagos – virtuális – meghajtót megszüntetjük)

(A **subst** használatának érdekes lehetősége, ha az **A:** betűjelet használjuk. Ekkor a floppymeghajtót nem használhatjuk, helyette a merevlemezen egy könyvtár lesz az **A:** egység. Ezzel egyes – floppy jelenlétét figyelő – programokat kicselezhetünk.)

Fontos: A billentyűzeten keresztül sokféle karaktert tudunk bevinni. Azonban nem mindegyiknek van külön gombja. Minden karakternek van egy 3 jegyű száma, ez az ún. ASCII kód. Segítségével a nem látható karaktereket – pl. ékezetes betűket – is be tudjuk vinni. Például az "é" kódja 130. A bevitelhez nyomva kell tartani valamelyik szürke Alt gombot, és le kell ütni a billentyűzet jobb oldalán található számbillentyűk közül az 1-et, majd a 3-at, végül a 0-t. Ezután az Alt felengedésekor a képernyőn megjelenik az "é" betű. Így lehet ékezetes karaktereket bevinni, azaz magyar nyelvű szövegeket létrehozni. (Pl. Alt+130="é", Alt+147="ő", Alt+148="ö", Alt+160="á" stb.) Az itt bemutatott lehetőség majdnem minden helyzetben működik (csak esetleg más karakterek jelennek meg). Próbáljuk ki!

Következő témakör a **futtatható fájlok** rövid áttekintése. Mint ahogy a feladatsor elején szó esett róla, egy fájl kiterjesztése utal a típusára. A fájlok közül a *futtathatók* **kiterjesztése háromféle lehet: .COM, .EXE és .BAT.** Tehát ha egy `dir` parancs kiadása során ilyen kiterjesztésű fájlokkal találkozunk, ezeket és csak ezeket indíthatjuk el. A kiterjesztések a COMmand=parancs, EXEcutable=futtatható és BATch=kötegfájl nevek rövidítéséből keletkeztek. **Ha egy alkönyvtárban azonos néven mindhárom fájl létezik és kiterjesztés nélkül kiadjuk a nevet, a számítógép a .COM fájlt fogja futtatni. Ha nincs, az .EXE-t, ha az sincs, a .BAT-ot** (a futtatás sorrendjére a "CEB" rövidítést szokták használni).

A `.COM` és `.EXE` fájlok a mikroprocesszor számára érthető, ún. gépi kódú programokat tartalmaznak. Számok sorozatából áll, kilistázva a fájl tartalmát (`type` parancs) értelmetlen karaktereket láthatunk. Ilyen fájlokat valamely programozási nyelv (Assembly, Pascal, C stb.) segítségével készíthetünk, amit a programcsomag lefordít gépi kódra.

A `.BAT` (kiejtve kb. "becs", *kötegfájlok*) fájlok olyan speciális állományok, amelyek *nem* gépi kódú utasításokból állnak, hanem DOS parancsok sorozatából. Ha néhány parancsot akarunk több alkalommal kiadni, érdemes batch fájlt írni rá. A batch fájlok készítése már lassan programozási feladat: a DOS-t "programozzuk". A lehetőségek természetesen korlátozottak, egy-egy programozási nyelv jóval többet tud. A UNIX-ban is lehet ilyet készíteni (shell script – forgatókönyvnek hívják).

Példa:

• (A `GYAK` alkönyvtárban vagyunk.) Készítsünk `LISTAZ` néven batch fájlt, amely kiírja az aktuális alkönyvtár könyvtárait és fájljait, egymástól elkülönítve és ABC sorrendben! (Ez a rövid batch program nem sok mindenre jó, csak egy példa akar lenni.)

Lépjünk be a szövegszerkesztő programba: "`edit listaz.bat`" és írjuk be az alábbi sorokat – lehetőleg használjunk magyar ékezetes betűket! :

```
@rem Könyvtártartalom megjelenítő batch fájl
@echo off
cls
echo A könyvtárban található alkönyvtárak ABC sor-
rendben:
echo.
dir|sort|find "<DIR>"|find ". "/v
echo.
echo A könyvtárban található fájlok ABC sorrendben:
echo.
dir|sort|find "."| find "<DIR>"/v
echo.
echo Vége.
echo.
```

A beírás után kilépés mentéssel: Alt+F, Kilépés (Exit) menüpont és két Enter.

Próbáljuk ki a batch fájlt! Ehhez a promptba a nevét kell beírni ("**listaz**") és Entert ütni.

Magyarázat: a **rem** utasítás után megjegyzéseket írhatunk, amelyek a program későbbi tanulmányozásánál lehetnek hasznos. A batch fájl minden végrehajtandó sort kiír a képernyőre. Ha a sor elejére a *kukac* jelet ("@" – Shift+2 angol kódlapnál) írjuk, nem fognak a sorok megjelenni. Minden sor elé nem írjuk ki, inkább az **echo off**-fal globálisan letiltjuk a "visszabeszélést". Az **echo** paranccsal szöveget írhatunk ki a képernyőre, az **echo.** egy üres sort ír ki. Utána a **dir** parancs eredményét ABC sorrendbe rendezzük, kiválasztjuk a "**<DIR>**"-t tartalmazó sorokat, de kihagyjuk ezek közül a "**.** "-seket (a **.** és **..** kiszűrése miatt – pont és szóköz!). Így kapjuk meg a könyvtárakat. A fájlokat hasonlóan: minden fájlt tartalmazó sorban előfordul a "**.**", viszont így a könyvtárak is bekerülnek, amelyeket kiszűrünk. Hogy hogyan lehetett megírni ezt a batch fájlt: **rem/?, echo/?, find/?.**

• Bonyolítás: a batch fájl indításakor adjunk meg egy könyvtárnevet, a listázás erre vonatkozzon! Ha nem adunk meg semmit, a program adjon hibaüzenetet!

Megoldás: "**edit listaz.bat**" és bővítsük ki. A kibővített program:

```
@rem Könyvtártartalom megjelenítő batch fájl
@echo off
if "%1"=="" goto hiba1
if not "%2"=="" goto hiba2
if exist listaz.txt goto hiba3
cls
echo A(z) %1 könyvtárban található alkönyvtárak ABC
sorrendben:>listaz.txt
echo.>>listaz.txt
dir %1|sort|find "<DIR>"|find ". "/v>>listaz.txt
echo.>>listaz.txt
echo A(z) %1 könyvtárban található fájlok ABC sor-
rendben:>>listaz.txt
echo.>>listaz.txt
dir %1|sort|find "."| find "<DIR>"/v>>listaz.txt
echo.>>listaz.txt
type listaz.txt|more
del listaz.txt
echo Vége.
echo.
goto end
```

```
:hiba1
echo Hiba! Kérek egy könyvtárnevet paraméterként! Pl.
"listaz c:\dos" !
goto end
:hiba2
echo Hiba! EGY paraméter kell!
goto end
:hiba3
echo Hiba! A batch fájl felhasznál egy ideiglenes
listaz.txt fájlt, ami
echo már létezik! Törölje le és utána próbálja újra!
:end
```

A beírás után lépjünk ki mentéssel (Alt+F, Kilépés sor és két Enter) és próbáljuk ki a batch fájlt paraméterrel és anélkül! Példák: `listaz` ; `listaz c:\dos` ; `listaz c:\win95` ; `listaz .` ; `listaz ..` ; `listaz c:\ c:\`.

Magyarázat: létrehozunk egy fájlt (>), amelybe átirányítással (>>) fokozatosan hozzáfűzzük a képernyőn megjelenítendő szöveget, végül a **type** és **more** parancsok segítségével oldalakra tördelve megjelenítjük. A paramétert a batch fájlban a `%1`-gyel tudjuk kezelni (`%2` a második paraméter stb.). Ha a paramétersor üres, vagy nem egy paramétert tartalmaz, vagy az ideiglenes fájl már létezik, hibajelzést adunk (`if` utasítás).

Ez már majdnem programozás! A Programozás tantárgyban tanult Turbo Pascal programok is hasonló elven készült sorokból épülnek fel!

• Ha egy programot több tíz floppyra szeretnénk felmásolni, az alábbi rövid batch fájl segíthet (ne gépeljük be, csak próbáljuk megérteni működését):

```
@rem Floppymásoló batch fájl
@echo off
cls
:vissza
echo Tegye be a nem írásvédett floppyt és üssön le
egy billentyűt!
pause>nul
echo Másolás indul...
copy munka.txt a:
choice /c:IN /n /t:N,10 Újra?
if errorlevel 2 goto vege
```

```
goto vissza
:vege
echo Vége.
echo.
```

Magyarázat: a **nul** a fekete lyuk, így a **pause** szövegét el tudjuk tüntetni. A **choice** és **if-errorlevel-goto** parancsokkal a billentyűzetről dönteni tudunk, hogy a program futása hol folytatódjon. Az **if** parancs működése miatt az **errorlevel**-t a nagyobbtól az alacsonyabb értékek felé kérdezzük le (**choice/?, if/?**).

- Próbáljuk megérteni a következő batch fájl működését, akár ki is próbálhatjuk:

```
@rem Kérdező batch fájl
@echo off
if exist semmi.txt goto hiba
:start
cls
echo Hogy érzed magad?
echo (Írd be: jól vagy rosszul, a végén F6 és
Enter)
copy con semmi.txt>nul
find /I "jól" semmi.txt>nul
if errorlevel 1 goto nincsjol
echo Jól? Örülök neki!
goto vege
:nincsjol
find /I "rosszul" semmi.txt>nul
if errorlevel 1 goto nincstalalat
echo Rosszul? Sajnálom!
goto vege
:nincstalalat
echo Hiba! Rossz választ írt be!
choice/N/T:I,2
goto start
:hiba
echo Hiba! A batch fájl használna egy semmi.txt-t,
ami már létezik!
:vege
del semmi.txt
```

Magyarázat: Mindjárt az elején megvizsgáljuk, hogy az általunk ideiglenesen használt fájl létezik-e. Ha igen, hibaüzenettel leállunk (mert különben felülírnánk – ez egy hibakezelés). A válasz egy fájlba kerül, de a felhasználónak erről nem kell tudnia. DOS-ban a fájl vége jel valójában Ctrl+Z (és Enter), de F6 (és Enter) is jó. A `find` beállítja az `errorlevel` értékét találattól függően, ezt lekérdezzük és ettől függően ugrunk, majd az `echo`-val kiírjuk a választ. A `>nul` hatására a find nem ír semmit a képernyőre (a zavaró kiírást elnyomjuk). A `choice`-t most időhúzásra (2 másodperc) használjuk. A program végén a `del` paranccsal letöröljük az ideiglenes `semmi.txt` állományunkat.

- Batch fájloknál használható még a `for` parancs is, amely egy adott parancsot a fájl csoport minden egyes tagjára végrehajt.

Példa: az aktuális alkönyvtárban keressük meg az összes `.TXT` fájlban a "számítógép" szót (→`for/?`):

```
for %f in (*.txt) do find "számítógép" %f
```

Példa: a teljes C: háttértáron, bármely fájlban keressük meg a "számítógép" szót:

```
dir c:\ /s /b | find "számítógép"
```

Következő témakör a `path` (kiejtve kb. "pász", angolul ösvény). **A DOS egy fájlt először az aktuális könyvtárban keres.** Most a gyökérkönyvtárban tartózkodva írja be a "`proba`" parancsot – hibaüzenetet fogunk kapni ("`Bad command or file name`"), a gép nem találja. A korábban használt `EDIT` szövegszerkesztő program a `C:\DOS` alkönyvtárban elhelyezkedő `EDIT.COM`, ezt hogy tudtuk elindítani a `GYAK`-ból? Megoldás **a path (elérési út).** Ez **egy DOS "változó", amely különféle könyvtárneveket tartalmaz.** Ha a keresett állomány nem található az aktuális könyvtárban, a gép a path alapján tovább keres. Amennyiben egyikben sem találja, akkor írja csak ki az említett hibaüzenetet.

- Nézzük meg a path változó tartalmát!

("`path`" – a pontosvesszővel elválasztott különféle könyvtárnevek között felfedezhetjük a `C:\DOS` vagy `C:\WIN95\COMMAND` bejegyzést, az `EDIT` programot így tudta megtalálni a gép.)

- Írassuk ki a DOS belső változóit!

("`set`" – a path-on kívül láthatjuk az ideiglenes, TEMPorary alkönyvtár nevét. A "`PROMPT=$P$G`" jelzi, hogy a prompt "`C:\>`" alakú legyen – teljesen más formákat is létrehozhatunk. Az alapforma "`C>`". A `TEMP` a temporary – ideiglenes könyvtár helyét jelzi. E változókat az `AUTOEXEC.BAT` fájlban állíthatjuk be, lásd később.)

- Töröljük le a `GYAK` alkönyvtárat összes tartalmával együtt! Ehhez először vissza kell lépni a főkönyvtárba.

("`cd\`" és "`deltree gyak`".)

*A DOS és a Windows rendszerek semmilyen védelmet nem nyújtanak: egy-egy megfelelően kiadott **del** vagy **deltree** paranccsal letörölhetjük magát az operációs rendszert! A védelem a Windows NT-ben, illetve a UNIX rendszerekben jelenik meg.*

A következőkben a `date` és `time` utasításokon túl ismerjünk meg még két karbantartó programot. Ezeket a hasznos segédprogramokat is érdemes havonta egyszer lefuttatni. Nevük `scandisk` és `defrag`.

A `scandisk` a háttértárak hibáival foglalkozik. Pl. ha a lemez egy része mechanikai hatás miatt megsérült, erre a területre nem célszerű adatokat írni. A `scandisk` megtalálja ezeket a részeket és "Bad sector" – rossz (hibás) szektornak bejelöli, kiiktatja, mentéskor nem kerül ide adat. Gyakorlatilag ha a merevlemezen, vagy egy hajlékonylemezen egyszer megjelenik a bad szektor, hamarosan egyre több lesz belőle, a "`Data error reading drive A`" üzenet elkerülése miatt érdemes azonnal a kukába dobni és márkás lemezt beszerezni. Ajánlható a 3M, a TDK, a Sony stb. A "noname" (névtelen) lemezek használata kakukktojás: vagy jobb lesz, mint egy márkás, vagy a második használatkor tönkremegy. Fontos anyagainkat a biztonság kedvéért érdemes márkás – drágább – adathordozón tárolni.

- A program indítása: "`scandisk`". Próbáljuk ki nyugodtan, majd menet közben lépjünk ki belőle! Windows esetén a program Windowsos változata indul el, kilépés után vissza kell váltani a DOS ablakba!

A **defrag** a széttöredezett fájlokat rendezi a lemezen fizikailag egybe. Ez mit jelent? Egy fájl mentésekor a DOS a háttértáron levő üres helyekre ment (logikus, nem?). Az üres helyek – egy-egy régebbi törlés miatt – nem folytonosan helyezkednek el, így a kimentett fájlunk is több részletben lesz megtalálható, más szóval fragmentált, széttöredezett lesz. (Ezzel nem kell törődni, a fájlrészletek helyét a DOS nyilvántartja, a széttöredezettségből nem látunk semmit.) A **defrag** kiadása után állományaink különböző részei, valamint az összes szabad terület fizikailag egybe kerül, így a fájlműveletek valamivel gyorsabbak lesznek.

- Indítása: "**defrag**". Ezt is próbáljuk ki, majd menet közben ebből is lépjünk ki! Windows esetén a kilépés után ismét vissza kell váltani a DOS ablakba!

Utolsó témakör a számítógép újraindítása. Előfordulhat, hogy egy program futás közben olyan utasításokra téved (programozói hibából, esetleg külső elektromos zavar hatására), hogy végtelen ciklusba kerül – a program lefagy, lemerevedik, kiakad. Elég nagy baj, mert valószínűleg nem tudunk kilépni DOS-ba, nem kapjuk vissza a promptot. Egy programból a Ctrl+Break, vagy a Ctrl+C billentyűkkel térhetünk vissza DOS-ba. Ha ez nem sikerül, újra kell indítani a gépet.

Kétféle újraindítás van. Az ún. ***melegindítás*** a billentyűzetről kezdeményezhető, a Ctrl+Alt+Delete billentyűzetkombinációval. Ekkor a gép újraindul, viszont nagyon sok dolog nem törlődik (interrupt vektorok, egyes vírusok stb.), komolyabb lefagyás esetén nem működik. A ***hidegindítás*** a számítógép házán található Reset gomb megnyomásával történik. Ez mindig működik, olyan, mintha egy pillanatra ki, majd bekapcsoltuk volna a gépet.

A számítógép újraindítása során az alaplapon található ROM FFFF0 hex (1048560 dec) címén kezdődő gépi kódú utasításokat kezdi a processzor végrehajtani. Itt egy ugró utasítás található, amely átirányít a ROM más részére (ez géptípusonként különböző cím lehet), ahol a tényleges program található. A **betöltő program** a kezdőértékek – interrupt vektorok stb. – beállítása után megvizsgálja, hogy található-e a rendszerben esetleges külső ROM. Amennyiben igen, a vezérlést átadja, így a hardver eszközök – videokártya stb. – hozzá tudnak kapcsolódni a rendszerhez. Ezek befejeztével a

program egy **RAM tesztet** végez (a képernyőn lehet látni, növekvő számsorozat formájában), majd a billentyűzet ellenőrzése következik – egy pillanatra felvillan a három LED -, végül a háttértárakat vizsgálja meg a gép. Ezután letörlődik a képernyő és (a legtöbb gépen) egy téglalapban megjelennek a gép hardver adatai. Befejezésül a **háttértáron operációs rendszert keres a rendszer.** Jelen esetben megtalálja pl. a DOS-t és betölti (**"Starting MS-DOS..."** felirat megjelenik). A betöltést boot-olásnak nevezzük (kiejtve kb. "bútolás", boot-ol, azaz a hardver építményre ráhúzza a vezérlő keret – shell – programot, mint ahogy a csizmát felhúzzuk a lábunkra), részletesen a jegyzet elején található. **A DOS boot-olása:** először betöltődnek a **C:\IO.SYS** és a **C:\MSDOS.SYS** rejtett fájlok és végrehajtódik a **C:\CONFIG.SYS** rendszerfájl, amely betölti a különféle memóriarezidens programokat. Ezután betöltődik a **COMMAND.COM** parancsértelmező, végül lefut a **C:\AUTOEXEC.BAT** batch fájl, mely a felhasználó igénye szerinti beállításokat – path, prompt, temp stb. – tartalmazza. Ezek után jelenik meg a prompt. A **CONFIG.SYS** és **AUTOEXEC**.BAT egyszerű szövegfájlok, amelyeket pl. az EDIT szövegszerkesztővel is módosíthatunk, általában minden DOS-os, vagy Windows-os számítógépen megtalálhatók. A **Starting MS-DOS...**, vagy **Starting Windows 95...** felirat megjelenésekor az F4, F5 és F8 billentyűk megnyomásával beavatkozhatunk a betöltés folyamatába. Pl. Windows 95/98-nál F8-ra egy menü jelenik meg, amelyből több üzemmód közül választhatunk, F4-re a régi (általában az angol 6.22-es) MS-DOS töltődik be.

- Nézzük meg a **config.sys** tartalmát!

("**type config.sys**". A listából a "**device=**", "**devicehigh=**", vagy "**install=**" szavak után egy-egy teljes elérési úttal megadott program szerepel, ezeket a rendszer betölti memóriarezidens programként.)

- Nézzük meg az **autoexec.bat** tartalmát is!

("**type autoexec.bat**" – (a legtöbb esetben) látható a "**@echo off**" és a "**path=pg**" sorok, amelyekről már volt szó.)

Ezzel véget ért a DOS gyakorló feladatsor. Lépjen ki és kapcsolja ki a számítógépet!

Az itt leírt anyag viszonylag tömény, teljes megértéséhez esetleg több részletben, több alkalommal szükséges végighaladni a feladatokon. Így elsajátítja a DOS alapjait annyira, hogy kézikönyvből önállóan meg tudja ismerni a teljes rendszert és a Windows használatakor sem ütközik majd gondokba.

Röviden még egyszer: az iménti feladatsor a DOS alapjainak begyakorlására szolgált. Magával a DOS-szal nem lehet játszani, hanem egy-egy adott feladat elvégzéséhez (pl. floppyról új program telepítése, munkánk másolása lemezre, biztonsági másolatok készítése, gépünk karbantartása stb.), egyszóval a napi munkához ismerni kell fejből néhány parancsot. Ezekről esett szó. Legfőképpen az elvet kellene megérteni, hogy más operációs rendszerrel is könnyen boldogulni tudjunk.

9.5. Miről nem esett szó?

Floppy és más meghajtók használata

A gyakorló feladatsorban végig a winchesteren dolgoztunk. Más háttértár használatához át kell váltani a megfelelő meghajtóra. A váltás úgy történik, hogy a prompt-ba beírjuk a meghajtó betűjelét *és egy kettőspontot.*

Pl. átváltás hajlékonylemezre: "**a:**" vagy "**b:**". A CD-k betűjele általában D, a hálózatoké pedig F-Z.

```
C:\>a:

A:\>_
```

Ha nincs lemez a meghajtóban, a gép a "**Not ready reading drive A, Abort, Retry, Fail?**" üzenetet írja ki. Ekkor az F betűt üssük le (magyarnál M-Mégse)! **A** "**Current drive is no longer valid>**" üzenetre pedig be kell írni a "**c:**"-ot, így tudunk visszaállni az előző meghajtóra.

Egyéb példák: **C:\>dir a:** , **C:\jatek>copy a:*.* .** , **A:\>del.** stb.

Formattálás

A hajlékonylemezeket és a merevlemezt első használat előtt formattálni kell, ekkor a gép a DOS formátumnak megfelelően felosztja logikai részekre. A merevlemezt ritkán szoktuk formattálni, a hajlékonylemezeket inkább, bár az utóbbi időben már lehet kapni gyárilag formattált ("*MS-DOS Formatted*") floppykat is, persze drágábban. A parancs formája: "**`format a:`**", "**`format b:`**" stb. Ez a művelet minden előző adatot töröl a lemezről! A különösen "veszélyes" és ritkán használt "**`format c:`**" parancsnál a gép még egy megerősítést vár. A formattálás előtti állapot az **`undelete`** parancshoz hasonlóan visszanyerhető, az **`unformat`** parancs kiadásával. Vírusos, vagy vírusgyanús lemezeket a "**`format a:/u`**" paranccsal szokták letörölni, a **`/u`** paraméter ténylegesen töröl, nem lehet utólag visszaállítani a lemez tartalmát.

Particionálás

Egy merevlemezt logikailag több részre feloszthatunk, más-más betűjelű meghajtókat hozhatunk létre, akár különböző operációs rendszereknek. Ezek egyikét kijelölhetjük aktívnak, innen fog a bootolás történni. Az ezt végző segédprogram az **`fdisk`**, amelyet csak kellő ismeret után használjunk, mivel a partíciók megváltoztatása törli a merevlemez tartalmát! (Egyes boot vírusokat kiírthatunk az "**`fdisk c:/mbr`**" nem publikált paraméterrel – az mbr a merevlemezen található *Master Boot Record* rövidítése.)

Alacsony szintű formázás

A merevlemezeket legeslegelőször a gyárban ún. alacsony szinten formattálják ("*low level format*"). Egyes alaplapok ROM-BIOS-ával, vagy külső segédprogramokkal is végezhető. *Bizonyos merevlemez típusoknál a házi alacsony szintű formázás a garancia elvesztésével jár és tilos!*

*Tehát egy merevlemez használatbavételének lépései: alacsony szintű formázás, particionálás (**`fdisk`**), formattálás (**`format`**), operációs rendszer telepítése (installálása – általában floppyról vagy CD-ről).*

CMOS Setup

A számítógép bekapcsolásakor a RAM teszt után általában a szürke *Delete* billentyű lenyomásával juthatunk be egy menübe. Itt állíthatjuk be a háttértár adatait és az alaplap jellemzőit. Beállíthatjuk azt is, hogy induláskor jelszót kérjen a gép, de ez nem sokat segít, mert egyrészt a gombelem kivételével a CMOS RAM tartalma törlődik, másrészt sok alaplap *Global password*-je közismert. A beállított adatokat a CMOS RAM tárolja. Fontos, hogy ez a program a ROM-BIOS-ban található, ezért akár merevlemez és operációs rendszer nélküli gépen is használható.

Rendszerlemez készítése

A rendszerlemez egy floppy, mely bootolható, azaz a gép bekapcsolásakor behelyezve elindul a – minimális – operációs rendszer. Akkor használják, ha a merevlemezen nincs op. rendszer, letörlődött, vagy vírusos lett.

Rendszerlemez készítése: **`C:\>format a:/s`** vagy: **`C:\>sys a:`**

Mindig legyen számítógépünk környékén egy rendszerlemez, sose tudhatjuk, mikor fog kelleni!

9.6. A DOS parancsok összefoglalása:

A parancs szintaktikáját és a felhasználható paramétereket a "**`parancsnév/?`**", vagy a "**`help parancsnév`**" kiadása után láthatjuk. A név után a B/K a belső/külső parancsot jelöli.

Emlékeztetőül: a belső parancsot a **`COMMAND.COM`** parancsértelmező ismeri fel és hajtja végre, míg a külső parancs a **`C:\DOS`**, illetve a **`C:\WIN95\COMMAND`** alkönyvtárban található **`.EXE`** vagy **`.COM`** kiterjesztésű segédprogram, amit a nevének kiadásával indítunk el, hasonlóan bármely más programhoz. A DOS alkönyvtár általában szerepel az **`AUTOEXEC.BAT`**-beli **`PATH`**-ban, így az elérési utat nem szükséges beírni.

`attrib`	K –	fájlok és könyvtárak jellemzőit (írásvédettség stb.) nézhetjük és változtathatjuk meg
`cd`	B –	aktuális könyvtár megváltoztatása
`cls`	B –	képernyő törlése (Clear Screen)
`copy`	B –	fájl másolása (még egy példány készítése) más könyvtárba, esetleg más néven. Ritkábban: állományok összefűzése, vagy új szövegfájl létrehozása.
`date`	B –	karbantartó parancs: a gép belső órájának (év-hónap-nap-napnév) kiírása és megváltoztatása
`defrag`	K –	karbantartó parancs: a háttértár állományait fizikailag folytonossá rendezi
`del`	B –	fájlok törlésére szolgál. **`del.`** = az alkönyvtár összes fájljának törlése
`deltree`	K –	teljes könyvtárakat töröl, tartalommal együtt. Veszélyes parancs!
`dir`	B –	könyvtár tartalmát jeleníti meg, fájlokat/könyvtárakat kereshetünk (`dir/s`)
`doskey`	K –	a prompt használatát megkönnyítő segédprogram
`echo`	B –	szövegek kiíratására szolgál (általában batch fájlokban).
`edit`	K –	a DOS szövegszerkesztő programja
`fc`	K –	két fájlt összehasonlíthatunk, megjeleníti a különbségeket
`fdisk`	K –	a merevlemez partícióinak beállítása. Vigyázat: a háttértár adatait megsemmisíthetjük!
`find`	K –	a bemenetben megadott szövegben szövegrészletet keres, az eredményt a kimenetre írja
`format`	K –	első használat előtt a háttértárat kialakítja a DOS formátumnak megfelelően
`graphics`	K –	rezidens program, grafikus üzemmódú képernyő kinyomtatását végzi el (aktiválása: PrintScreen)
`md`	B –	könyvtár létrehozására szolgál (az aktuális tartózkodási helyből fog nyílni)
`mem`	K –	számítógép memóriájának felosztása, rezidens programok felsorolása
`more`	K –	a bemeneti információt oldalakra törve írja ki a képernyőre
`move`	K –	fájlok mozgatására szolgál. A `copy`-tól annyiban tér el, hogy az eredeti állományt le is törli, de az áthelyezést csak logikailag (a nyilvántartásban) végzi el. Másik haszn: könyvtár átnevezése.

path	B –	elérési út megtekintése, megváltoztatása. Az **AUTOEXEC.BAT**-ban használják.
pause	B –	batch fájlokban vár egy billentyű leütésére
print	K –	szövegfájlok kinyomtatására szolgál
rd	B –	könyvtár megszüntetése (ha üres és nem tartózkodunk benne). Lásd még **deltree**.
rem	B –	batch fájlokban megjegyzések (remark) elhelyezésére szolgál
ren	B –	fájl átnevezése (könyvtár átnevezése: **move**)
scandisk	K –	karbantartó parancs: a háttértár hibáinak felfedezése, megjelölése és esetleg kijavítása
set	B –	DOS belső változóinak (**path, prompt, temp** stb.) kiírása
sort	K –	a bemeneten megadott információt bizonyos szempontok szerint rendezi a kimenetre (pl. ABC-be)
subst	K –	egy könyvtárhoz meghajtó betűjelet rendelhetünk (akár az **A:** -t is lehet!)
time	B –	karbantartó parancs: a gép belső órájának (óra-perc-mp-szmp) kiírása és megváltoztatása
type	B –	állományok tartalmának megtekintése. Általában szövegfájlokra használják.
undelete	K –	a **del**-lel letörölt fájlok visszaállítása (ha a háttértáron még megvan és nem **copy**-val írtunk felül létező állományt)
unformat	K –	formattált háttértár előző állapotának visszaállítása (**format/u** esetén nem működik)
ver	B –	a DOS verziószámának kiírása

A témakörhöz kapcsolódó kiadványok:

- Móricz Attila: DOS alapismeretek, LSI Oktatóközpont, Budapest 1998.
- Dallos-Fábián-Zombor: MS-DOS 6, LSI Oktatóközpont, Budapest 1993.
- Rozgonyi-Borus Ferenc: RAM-ba zárt világ, OSKAR kiadó Szombathely 1995.
- Daniel Ichbiah: Bill Gates és a Microsoft regénye, K.u.K. kiadó, 1996. (történelmi visszatekintés)

10. Commander programok

Bár a DOS minden lehetőséget megad a háttértár állományainak kezelésére, a fájlműveletek a parancsok begépelése miatt viszonylag nehézkesek. Ezért fejlesztették ki a különböző segédprogramokat, amelyek a fájlműveletek végrehajtását meggyorsítják és barátságosabb környezetet biztosítanak.

Ezek a programok tehát *nem* részei az operációs rendszernek, külön kell őket beszerezni. Közülük a legrégebbi és legismertebb a Norton Commander, mely hasonló karriert futott be, mint az MS-DOS, vagy a Windows. Az NC mellett megjelentek a 99%-ban ugyanolyan tudású, de ***shareware*** (bizonyos ideig – általában egy hónapig – szabadon kipróbálható a program, utána döntenünk kell: megvásároljuk a regisztrált változatot vagy letöröljük; más nevük *evaluation copy*) és ***freeware*** (jelentése általában: készítője minden jogról lemondott, szabadon terjeszthető és felhasználható) vagy *public domain* (mint az előző, bizonyos megkötésekkel) segédprogramok:

- Volkov Commander
- DOS Navigator
- Far manager
- Midnight Commander (Linux alatt)

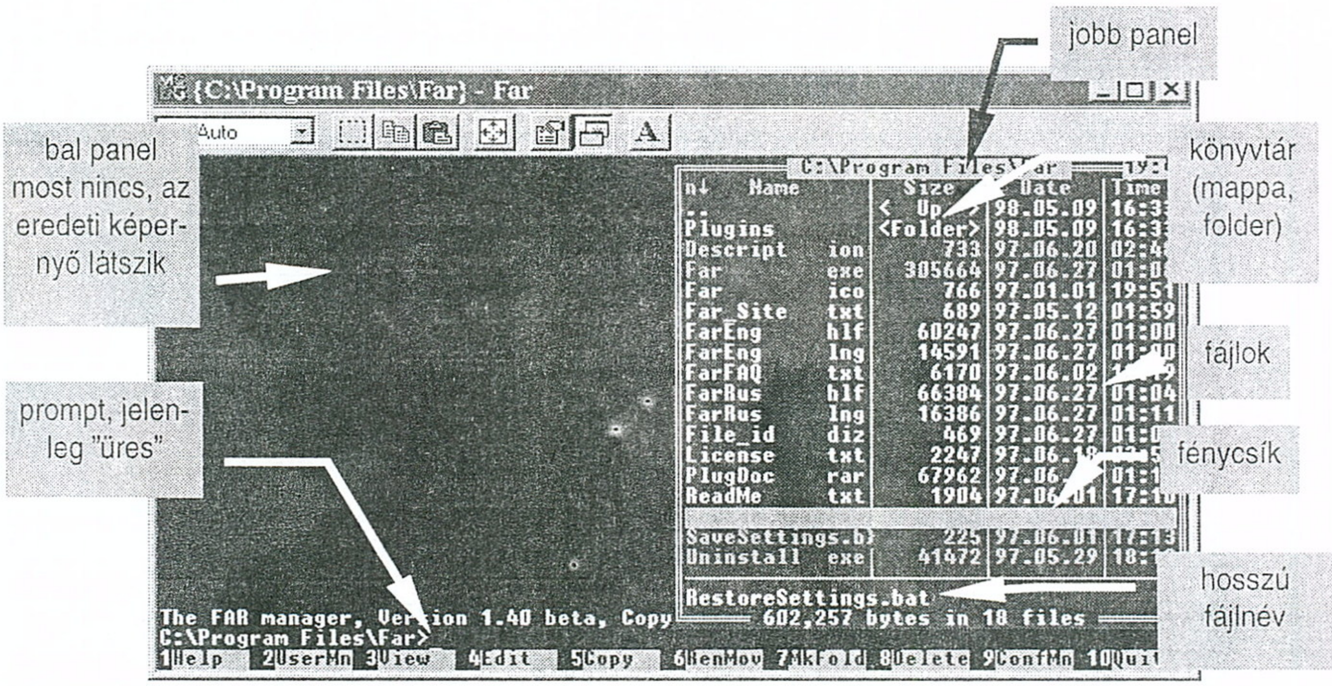
A Windows grafikus felhasználói felület elterjedése sem csökkentette e programok népszerűségét: sokan használják a Windows Intéző helyett valamelyik Commander régi DOS-os, vagy Windows alatti változatát. (Sőt, sokan nem is tudják nélkülözni: ha nincs ilyen a gépen, egy fájlmásolást sem tudnak végrehajtani.)

Ebben a fejezetben az egyik ilyen elterjedt segédszoftver, a FAR manageren keresztül tekintsük át a Commander programok kezelését.

A FAR manager Windows 95/98 és NT alatt futó program. Tulajdonságai egyebek között:

- kezeli a hosszú fájlneveket,
- a törölt fájlok a Lomtárba (Recycle Bin) kerülnek,
- tömörített állományok belsejével is lehet dolgozni.

A Commander programok lényege: a DOS képernyő és a prompt megmaradnak és használhatók, ezen felül a képernyő "előtt" két ún. *panel* jelenik meg, amelyek a plusz szolgáltatásokat biztosítják:



A prompt ugyanúgy használható, mint régebben, azaz tetszőleges DOS parancsokat adhatunk ki: programokat indíthatunk stb. A paneleken mindig az aktuális könyvtár tartalma látható (alkönyvtárak és fájlok), a DOS **`dir`** *parancsához hasonlóan.*

A következő panelműveletek szinte mindegyik Commandernél ugyanúgy működnek:

Billentyű	*Jelentése*
Ctrl+F1, Ctrl+F2	bal/jobb panel megjelenítése és eltüntetése
Alt+F1, Alt+F2	bal/jobb panel meghajtó váltás (`A:`, `C:` stb.)
Ctrl+O	mindkét panel eltüntetése/megjelenítése
Ctrl+U	bal és jobb panel felcserélése
Ctrl+R	panel tartalmának újraolvasása (Re-read)
Tab	a "fénycsík" áthelyezése egyik panelről a másikra
kurzor fel és le	fénycsík mozgatása panelen belül; ha nincs panel, akkor a DOSKEY-hez hasonló funkció: az előző parancsok megjelenítése és beillesztése a promptba
Home, End	fénycsík mozgatása a panel legtetejére/legaljára
kurzor balra és jobbra	a promptba írt szöveg szerkesztése
Enter	• ha a fénycsík könyvtáron áll: belépés könyvtárba, illetve vissza a szülőkönyvtárba • ha futtatható fájlon áll: program elindítása • De! Ha a prompt nem üres, hanem karaktereket tartalmaz: a beírt parancs(ok) "kiadása", elküldése.
Ctrl+Enter	a fénycsík alatt álló fájlnév bemásolása a promptba, a kurzor helyétől
Insert	A fénycsíkkal csak egy fájlt tudunk kijelölni. Több fájl kijelölése és a kijelölés megszüntetése az Insert gombbal történhet (pl. másolás vagy törlés esetén).
szürke +	több fájl kijelölése (pl. *.*, `*.txt` stb.)
szürke -	a szürke + kijelölések megszüntetése
szürke *	a szürke + -szal kijelölt és a nem kijelölt fájlok (kijelölés szempontjából történő) felcserélése
Alt+F7	fájl keresése a háttértáron úgy, hogy megadjuk tartalmának egy részletét – karaktersorozatot (!!!)
Alt+F10	könyvtárváltás a fastruktúra segítségével
Ctrl+L	rövid információ megjelenítése a háttértárról

A Commander programok panelműveletei

A képernyő legalsó sorában további parancsokat láthatunk, egérkattintással, vagy a funkcióbillentyűk lenyomásával aktiválhatjuk.

A következő parancsok szintén majdnem mindegyik Commandernél egységesen használhatók:

Billentyű	*Jelentése*	*Megfelelő DOS parancs*
F1	a programról rövid információ megjelenítése (Help)	-
F2	felhasználói menürendszer megjelenítése (User Menu)	-
F3 vagy Alt+F3	fájl tartalmának megtekintése (View)	`type`
F4	létező (szöveg)fájl szerkesztése (Edit) Ezen belül: F2 – mentés (Save) F7 – string keresése (Search) Ctrl+Y – sor törlése Esc vagy F10 – kilépés (Quit)	`edit`
Shift+F4	*új* (szöveg)fájl létrehozása (Edit)	`edit`
F5	fájl, fájlok vagy könyvtárak másolása (többszörözése)	`copy`
F6	fájl, fájlok vagy könyvtárak mozgatása (áthelyezése), vagy: fájl vagy könyvtár átnevezése (RenMov)	`move` `ren`
F7	könyvtár (mappa, folder) létrehozása (MkDir)	`md`
F8	fájl, fájlok vagy könyvtárstruktúra törlése (Delete)	`del` `deltree`
F9	a képernyő tetején megjelenik egy újabb menü, amelyben különféle beállításokat végezhetünk el (programtól függően más és mást lehet) (PullDn)	-
F10	kilépés a Commanderből (Quit)	-

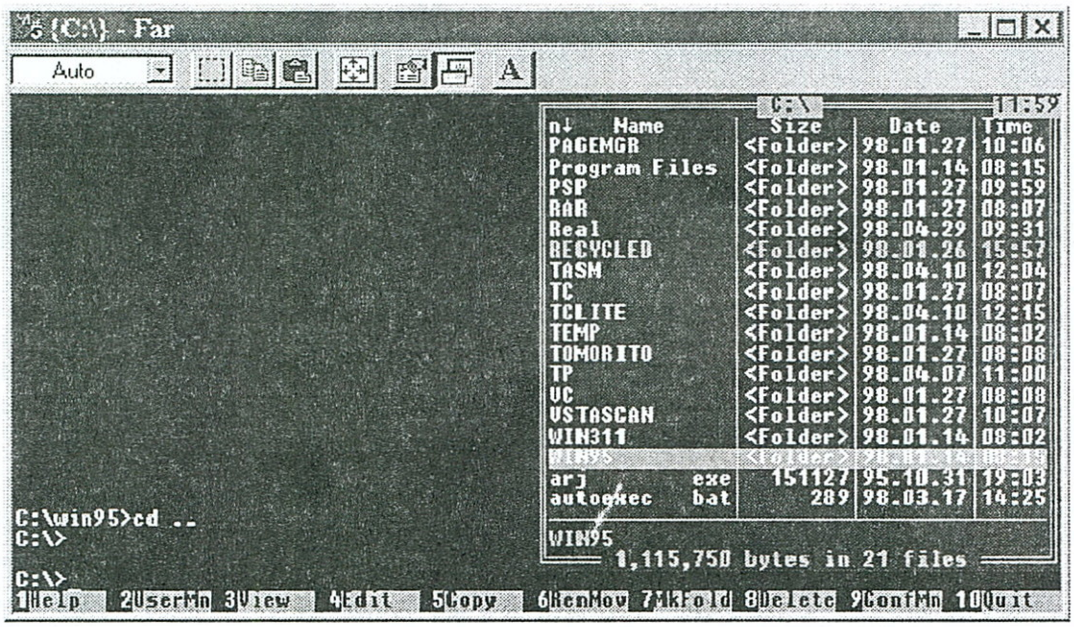
A Commander programok fájlműveletei

Példa:

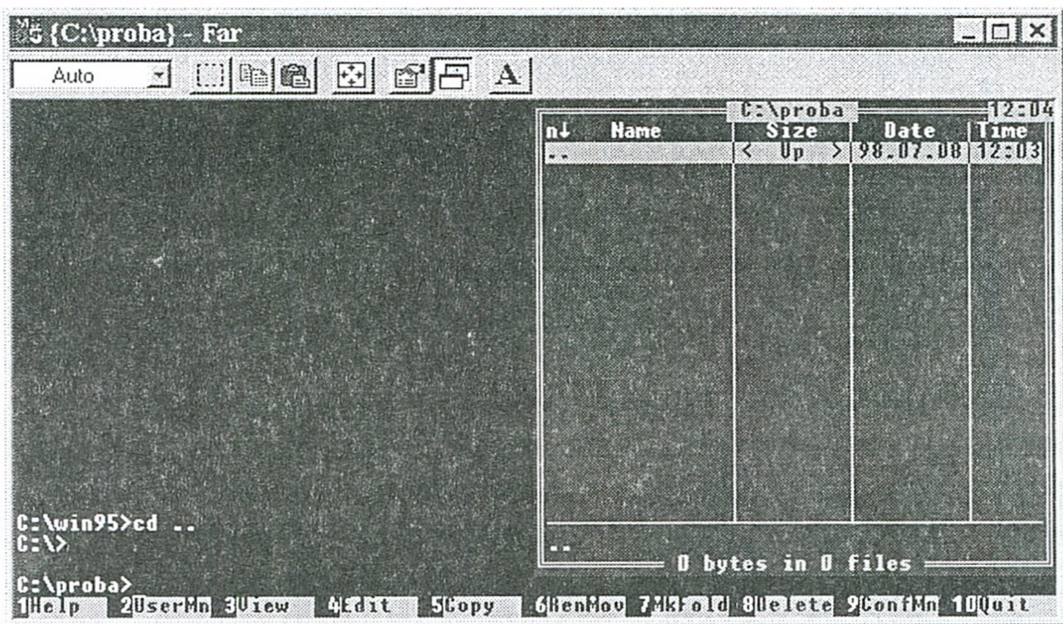
Hozzunk létre a gyökérkönyvtárban egy **`PROBA`** nevű alkönyvtárat, ide másoljuk át a **`COMMAND.COM`**-ot, nézzük meg tartalmát, nevezzük át, végül töröljük le a **`PROBA`** könyvtárat!

A lépések a következők:

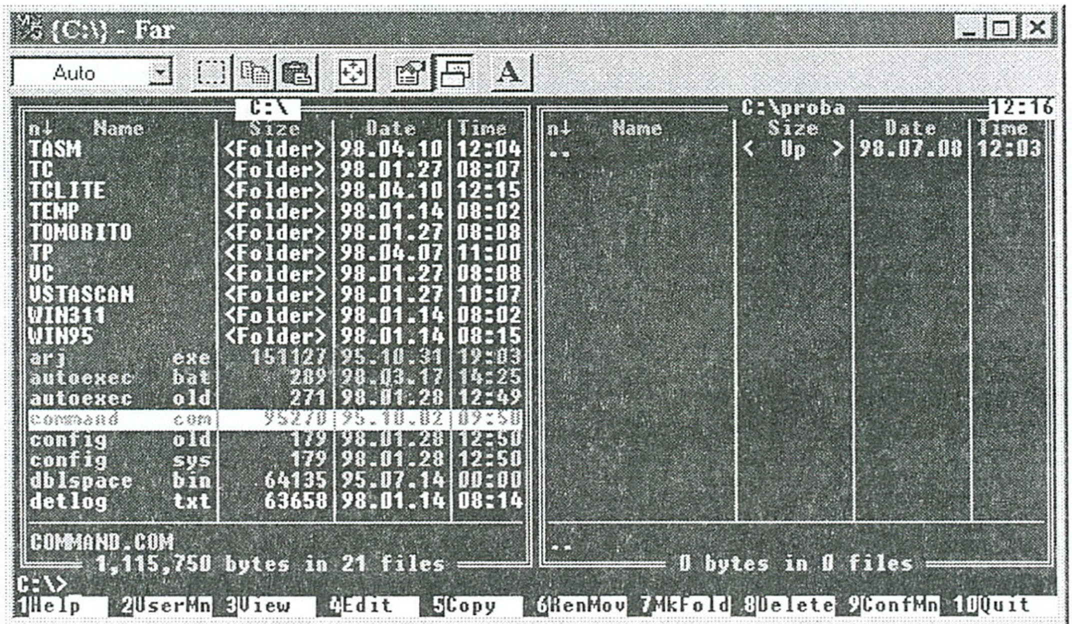
- Indítsuk el a (Norton, Volkov stb.) Commandert! A bal panelt a Ctrl+F1 esetleges megnyomásával tüntessük el, a jobb panelt a Ctrl+F2-vel jelenítsük meg! Ha nem állunk a gyökérkönyvtárban (a bal alsó sarokban a prompt nem "**`C:\>`**"-t mutat), a promptba adjuk ki a "**`cd ..`**" és Enter utasításokat addig, amíg ki nem jutunk a gyökérbe. A kiinduló helyzet:



• Most üssük le az F7-et, írjuk be **proba** (kisbetűkkel) és üssünk Entert. Létrehoztuk az alkönyvtárat. A fénycsík pont rajta áll és a promptba nem írtunk semmit, azaz ha most Entert ütünk, belépünk a – jelenleg üres – **PROBA** alkönyvtárba. Ezt egyrészt a bal alsó sarokban a prompt megváltozása, másrészt a panel teteje is mutatja:



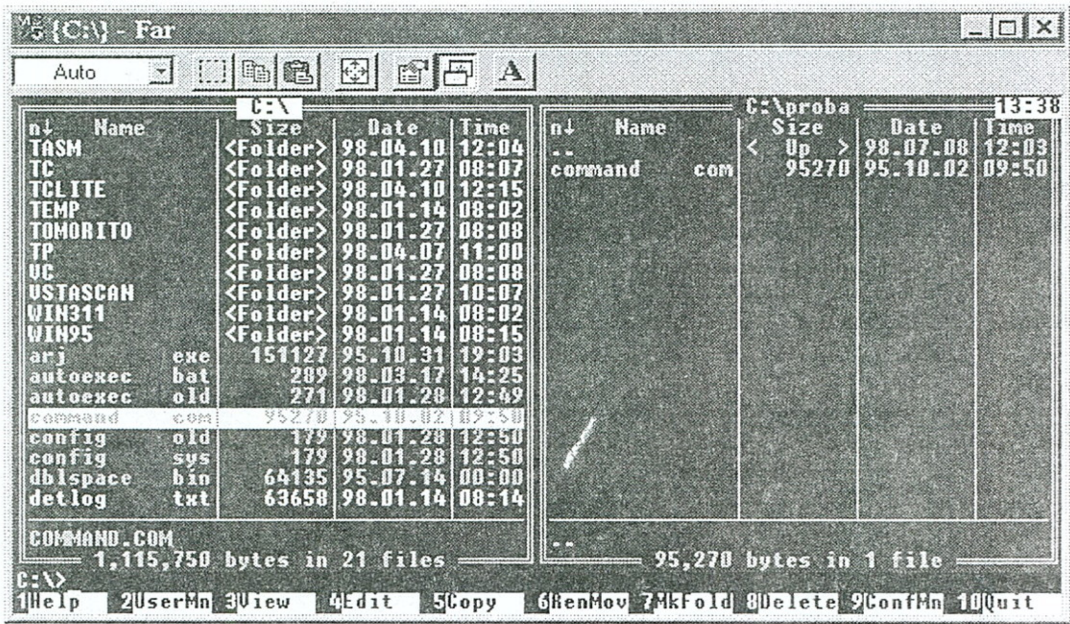
- Ide kell átmásolni a gyökérkönyvtárban található **COMMAND.COM**-ot, amit meg kell keresnünk, mégpedig a bal panelen. A Ctrl+F1-gyel jelenítsük meg a bal panelt! A panel tetejére van írva az aktuális meghajtó és könyvtár. Ha nem a **C:** meghajtón állunk, üssük le az Alt+F1-et és a C betűt! A **C:** egységre állás után fel kell menni a gyökérkönyvtárba. Ha nem vagyunk ott, üssük le a Home billentyűt és az Entert, ezzel egy könyvtárat visszalépünk. Ezt addig végezzük, amíg a gyökérkönyvtárba nem jutunk. Ezután a fel és le nyíllal addig lépkedjünk, amíg a "fénycsík" a **COMMAND.COM**-ra nem áll:



Ezzel a fájl másolását előkészítettük: az egyik panel (jelen esetben a bal) az átmásolandó fájl könyvtárát mutatja, a másik panel (most a jobb) a célkönyvtárat és a fénycsík az átmásolandó fájlon áll!

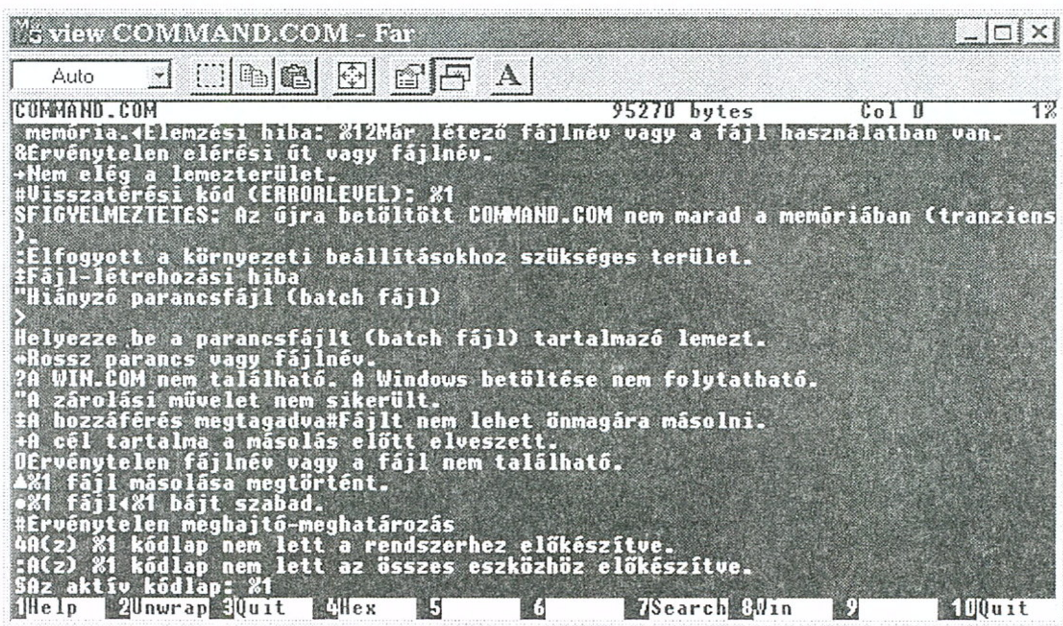
(Ha több fájlt kellene másolni, a fénycsík nem elegendő, az Insert gombbal kell a kívánt állományokat kijelölni.)

Üssük le az F5-öt és Entert, a másolás megtörténik:



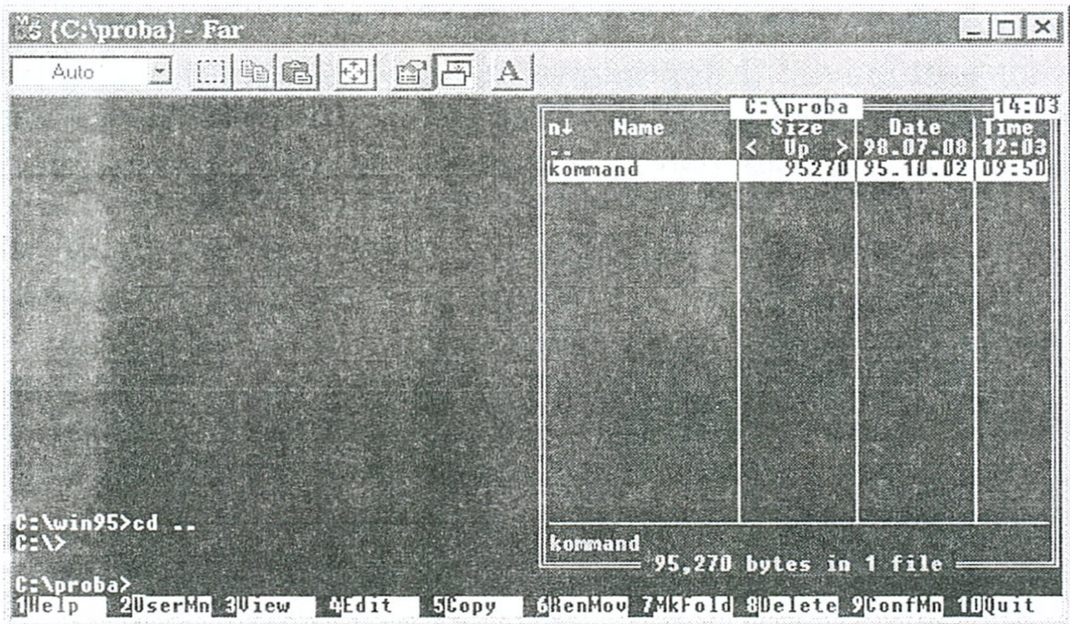
(Ha F6-ot ütünk, másolás – Copy – helyett mozgatást – Move – végeznénk, amit nem javaslok, mert a parancsértelmező nem véletlenül van a gyökérkönyvtárban.)

- Most nézzük meg a fájl tartalmát! A bal panelt Ctrl+F1-gyel rejtsük el (a fénycsík automatikusan átáll a jobb panelra), a le nyíllal vigyük a fénycsíkot a fájlnévre és üssünk F3-at (ha nem történik semmi, Alt+F3-at)! A fájl tartalmát láthatjuk (futtatható program, emiatt kriksz-krakszot). A további megtekintéshez a PageDown-nal lapozhatunk:

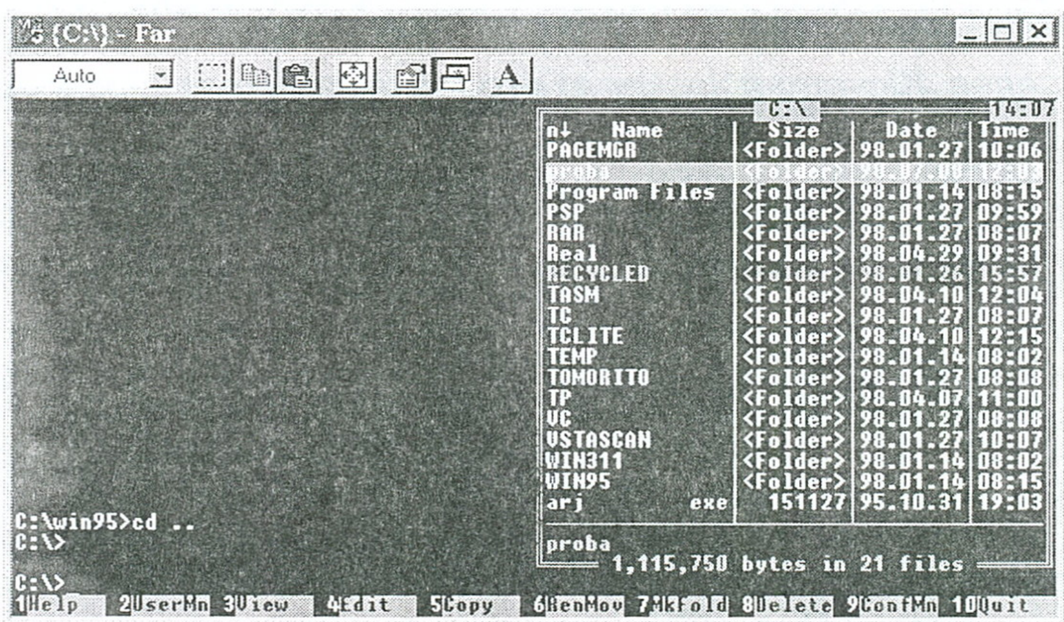


• Nevezzük át a fájlt! Először Esc-pel lépjünk ki a "View"-ból! *A fénycsík a fájlon áll és csak egy panel van a képernyőn* → minden készen áll a fájl átnevezésére!

Az átnevezéshez üssük le az F6-ot, írjunk be egy nevet (esetleg pontot és kiterjesztést), a végén Entert:



• Végül szüntessük meg a **PROBA** könyvtárat teljes tartalmával együtt! A Home és Enter billentyű segítségével kilépünk az alkönyvtárból a szülő könyvtárba, azaz a gyökérbe ("root directory"):



(A fénycsík a megfelelő helyen áll, a törlendő könyvtár nevén.) A könyvtár törléséhez az F8 és két Enter leütésére van szükség: az első Enter kiadja a törlés parancsot, a második egy megerősítés – tényleg törölni akarunk! A teljes alkönyvtár megszűnik (FAR manager esetén a Windows Lomtárba került), úgyhogy ez egy veszélyes művelet, hasonlóan a DOS `deltree` parancsához!

- A gyakorlás befejezéseként lépjünk ki a Commanderből az F10 és Enter leütésével! (Kilépni lehet csak F10-zel is, ha az F9-re megjelenő menürendszerben megkeressük a "Confirmations" sort és az "Exit" melletti bejelölést megszüntetjük – a módszer programonként más és más!)

E segédprogramokkal számtalan más művelet is elvégezhető, a fenti példa csupán egy rövid ízelítőt adott a lehetőségekből.

11. Tömörítő programok

A tömörítő programok fénykorukat a 80-as években élték, amikor a merevlemezek kapacitása még 100 KB-okban volt mérhető. Ha sok programot telepítettünk fel a számítógépünkre, előbb-utóbb megtelt a winchester és nem tudtunk tovább dolgozni.

A tömörítő programok a különböző fájlokat matematikai módszerekkel (pl. az ismétlődések rövidítésével) ***veszteségmentesen*** *tömörítik. A tömörített fájl mérete kisebb, így tárolásához kevesebb hely szükséges. Kicsomagoláskor az eredeti fájltartalmat kapjuk vissza.*

(Léteznek nagy hatékonyságú veszteséges tömörítési eljárások is, amelyeket a multimédiában filmek rögzítésénél használnak fel. Pl. a Sony Minidisc és a `.JPG` grafikus fájlformátum is veszteséges.)

A tömörítésnek két módja alakult ki:

- "**röptömörítés**": a winchesterre minden fájl tömörítve kerül fel, amikor egy fájllal dolgozunk, a rendszer előbb kicsomagolja és utána kerül át hozzánk. Mindez "láthatatlanul", a háttérben zajlik le, semmit nem tapasztalunk, a géppel ugyanúgy dolgozhatunk, mint korábban. Ezt a lehetőséget a 6-os MS-DOS a Doublespace/DriveSpace prog-

rammal támogatta, de elterjedtek más cégek szoftverei is, pl. a Stacker. A tömörített meghajtóra ilyen módon kb. 50%-kal több anyag fért fel, viszont az elérési idő kb. duplájára nőtt. A jelenlegi viszonylag olcsó, sok GB-os merevlemezek korában a röptömörítés lassan a háttérbe szorul.

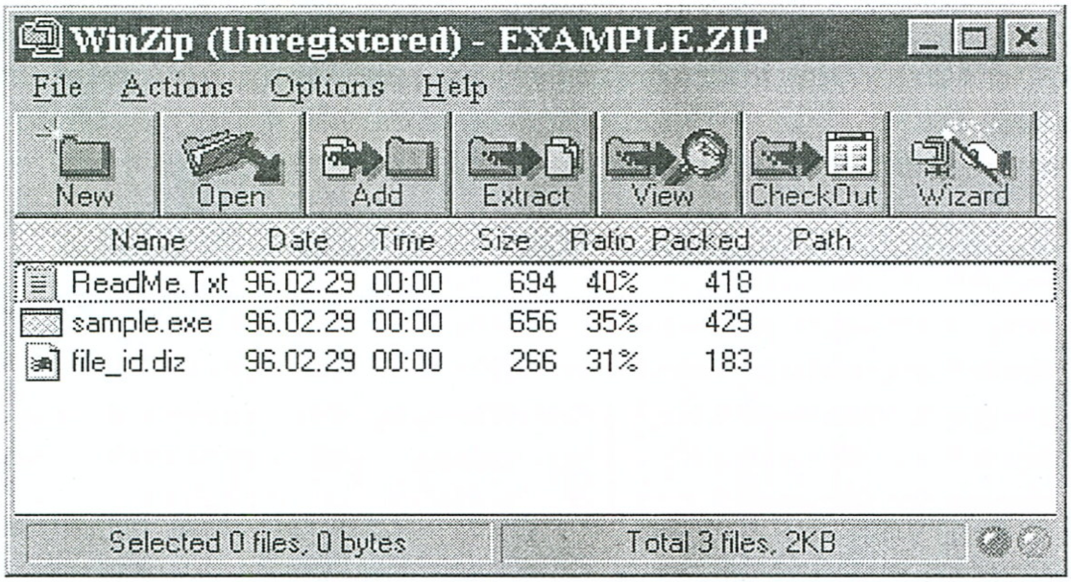
- **archiválás**: a megfelelő tömörítő programnak kiadjuk a parancsot, hogy bizonyos fájlokat tömörítsen össze, a végeredményt elmentjük floppyra és így tároljuk. Amikor szükségünk van a fájljainkra, előkeressük a floppyt és tartalmát kitömörítjük a merevlemezre. Ez a tömörítő programok régebbi és jelenlegi felhasználási köre. A CD-ken ”install” formában található programok szinte kivétel nélkül tömörített állapotúak, így kevesebb helyet foglalnak el, a telepítés során történik a kicsomagolásuk.

A továbbiakban csak a második csoporttal foglalkozunk. Hasonlóan a Commander programokhoz, *nem* részei az operációs rendszernek, külön kell őket beszerezni. A legelterjedtebb programok az alábbiak:

Program neve	Tömörített fájl kiterjesztése
`ARJ.EXE`	`.ARJ`
`PKZIP.EXE` `PKUNZIP.EXE`	`.ZIP`
`RAR.EXE`	`.RAR`
`LHA.EXE`	`.LZH`

A fájlok kiterjesztése utal tartalmára, tehát pl. egy **`.ARJ`** kiterjesztésű állomány nem futtatható, hanem az ARJ segédprogrammal betömörített fájlok sokaságát tartalmazza.

Az ARJ az egyik legrégebbi segédprogram. A **`.ZIP`** kiterjesztésű állományokkal különösen az Interneten, fájlletöltések alkalmával találkozhatunk. A RAR viszonylag új fejlesztés. Ezeknek az eredetileg DOS-os programoknak megjelent a Windowsos változata is, pl. az alábbiakban látható **`WINZIP`**:



☝ *Fontos, hogy a tömörített állománnyal dolgozni – speciális eseteket kivéve – nem lehet, csak archiválni. A munkához előbb ki kell tömöríteni!*

A tömörítők lehetőségeit az ARJ program segítségével tekintjük át.

Legfontosabb tulajdonság, hogy a különböző típusú fájlokat milyen mértékben tudjuk tömöríteni. Nézzünk néhány példát:

Fájltípus	Konkrét példa *	A fájl eredeti mérete	Fájlméret ARJ-vel tömörítés után	Tömörítési arány
program	**COMMAND.COM**	95.270 bájt	42.461 bájt	45%
grafikus kép	**FELHŐK.BMP**	307.514 bájt	110.432 bájt	36%
Word dokumentum	**HELLO95.DOC**	11.264 bájt	4.300 bájt	38%
Word dokumentum	**HELLO97.DOC**	19.456 bájt	1.632 bájt	8%

* A **COMMAND.COM** a Windows 95 parancsértelmezője, a **FELHOK.BMP** a rendszerhez tartozó fájl, a **HELLO95.DOC** és **HELLO97.DOC** Microsoft Word for Windows 95 és 97 változattal készült és csupán a ”hello” szót tartalmazza.

A futtatható programokat 40-100%-ra lehet tömöríteni (100% = nem sikerült tömöríteni). A **.BMP** grafikák és a **.DOC** Word dokumentumok viszont jól tömöríthetők (más oldalról megközelítve: e fájlok felépítése helytakarékosság szempontjából nem a legoptimálisabb).

Példa:

Az ARJ program segítségével nézzük meg a tömörítők által nyújtott szolgáltatásokat! A példák többségében egy fájlt, a **DEMO.TXT**-t tömörítjük be, az archívum neve pedig **ARCH** lesz.

Művelet megnevezése	*Kiadandó parancs (DOS-ból)*	*Keletkező fájl(ok)*
a program lehetőségeinek kilistázása	`arj/?`	-
egy fájl archiválása (tömörítése)	`arj a arch demo.txt`	`arch.arj`
két fájl tömörítése	`arj a arch demo1.txt demo2.txt`	`arch.arj`
egy alkönyvtár összes fájljának tömörítése	`arj a arch *.*`	`arch.arj`
tömörítés a könyvtárstruktúra megőrzésével	`arj a -r arch C:\munka\*.*`	`arch.arj`
archívum tartalmának megtekintése	`arj l arch\|more`	-
archívum épségének ellenőrzése	`arj t arch`	-
önkicsomagoló fájl készítése ellenőrzéssel	`arj a -je -jt arch demo.txt`	`arch.exe` (!!!)
túlméretes fájl több részre bontása (pl. hogy több floppyra ráférjen)	`arj a -v1440 arch demo.txt`	`arch.arj` `arch.a01` `arch.a02` **stb.** mindegyik mérete max. 1.44 MB!
kitömörítés	`arj x arch`	egy vagy több kicsomagolt fájl
archívumból *egy* fájl kitömörítése	`arj e arch demo.txt`	`demo.txt`

Az ARJ tömörítő program használata

(A több száz lehetőség közül csak a leggyakrabban használtakat néztük meg.)

Példa:

```
C:\TEMP>arj a -je -jt arch# demo.txt
ARJ 2.50 SHAREWARE Copyright (c) 1990-95 ARJ Software.
Oct 31 1995
*** This SHAREWARE program is NOT REGISTERED for use in
a business, commercial,
*** government, or institutional environment except for
evaluation purposes.

Creating self-extractor: ARCH#.EXE
Adding DEMO.TXT 100.0%
Testing DEMO.TXT OK
Self-extractor created.
 1 file(s)

C:\TEMP>arch#/l
ARJSFX 2.50 - ARJ Archive Self-Extractor. ARCH#.EXE -?
for help.
Copyright (c) 1990-95 ARJ Software. All Rights Reserved.

Processing archive: C:\TEMP\ARCH#.EXE
Archive date : 1998-07-08 17:48:26
Filename Original Compressed Ratio DateTime modified
CRC-32 Attr TPMG
------------ ---------- ---------- ----- --------------
DEMO.TXT 5 5 1.000 98-07-08 17:22:50 3610A686 A--W B 0
------------ ---------- ---------- -----
 1 files 5 5 1.000

*** NOT LICENSED for distribution use ***

C:\TEMP>del demo.txt

C:\TEMP>arch#
ARJSFX 2.50 - ARJ Archive Self-Extractor. ARCH#.EXE -?
for help.
Copyright (c) 1990-95 ARJ Software. All Rights Reserved.
Processing archive: C:\TEMP\ARCH#.EXE
Archive date : 1998-07-08 17:48:26
Continue extraction? y
Extracting DEMO.TXT, OK
 1 file(s)

*** NOT LICENSED for distribution use ***

C:\TEMP>_
```

Példa önkicsomagoló ("self extracting") fájl készítésére. Az archív fájl mérete valamivel megnő,

kibontásához viszont nincs szükség a tömörítő programra, egyszerűen el kell indítani.

A be- és kitömörítéshez nézzünk egy másik példát: készítsünk `.ZIP` archívumot! A tömörítéshez a `PKZIP.EXE`, a kicsomagoláshoz a `PKUNZIP.EXE` programot használhatjuk fel:

```
C:\TEMP>pkzip arch demo.txt

PKZIP (R) FAST! Create/Update Utility Version 2.04g 02-01-93
Copr. 1989-1993 PKWARE Inc. All Rights Reserved.
Shareware Version
PKZIP Reg. U.S. Pat. and Tm. Off. Patent No. 5,051,745

• 80486 CPU detected.
• EMS version 4.00 detected.
• XMS version 3.00 detected.
• Novell Netware version 4.11 detected.
• DPMI version 0.90 detected.
• Using Normal Compression.
Creating ZIP: ARCH.ZIP
 Adding: DEMO.TXT Storing ( 0%), done.

C:\TEMP>del demo.txt

C:\TEMP>pkunzip -d arch

PKUNZIP (R) FAST! Extract Utility Version 2.04g 02-01-93
Copr. 1989-1993 PKWARE Inc. All Rights Reserved.
Shareware Version
PKUNZIP Reg. U.S. Pat. and Tm. Off.

• 80486 CPU detected.
• EMS version 4.00 detected.
• XMS version 3.00 detected.
• DPMI version 0.90 detected.
   Searching ZIP: ARCH.ZIP
    Extracting: DEMO.TXT

   C:\TEMP>_
```

Egyszerű példa .ZIP archívum készítésére

12. Víruskereső és -irtó programok

A számítógépvírusokat olyan programozók készítik, akik a számítógép felépítéséről, az operációs rendszerről és az Assembly nyelvről az átlagosnál jóval többet tudnak, pl. ismerik az ún. "nem publikált" programhívásokat, az egyes rendszerekben található kiskapukat, "lyukakat", trükköket stb.

A számítógépvírus tehát mesterséges. Általában Assembly nyelven írják, mérete így nagyon rövid (néhány ezer bájt).

A 80-as évek magnókazettáival működő Sinclair számítógépekben nemigen lehetett vírust készíteni. Az IBM PC-ken viszont a kezdeti kazettás egység mellett megjelent a hajlékonylemez-, majd a merevlemez meghajtó. A háttértár fájlműveleteit a felhasználó beavatkozása nélkül a program önállóan intézi. Ez megteremtette a vírusok melegágyát.

A vírusfertőzés lépései az alábbiak:

1. ***Fertőzés.*** A fertőzés történhet úgy, hogy elindítunk egy fertőzött programot (azaz valójában a vírust indítjuk el), vagy ha vírusos lemezről boot-olunk. Mindkét esetben a vírusra kerül a vezérlés. Beül a memóriába ("aktív" lesz) és utána továbbadja a vezérlést az elindított programnak. Sajnos kaphatunk vírust "hivatalos" helyen vásárolt program esetén is!

2. ***Terjedés.*** Ezután hosszú ideig (napokig) nem veszünk észre semmit, a vírus közben észrevétlenül megfertőzi a merevlemezt, az operációs rendszert és minden fájlt, hajlékonylemezt, hálózatot, amihez hozzáfér. Közben *esetleg* időnként kiír egy-két üzenetet (pl. a One Half-fal fertőzött gép indításakor a "Starting MS-DOS" felirat *előtt* (!) néha kiírja *a "Dis is one half."* üzenetet; a Stoned vírus a *"Your computer is stoned."* feliratot írja ki néha). Az ügyesebb vírusok természetesen elrejtik pl. a `dir` parancs elől a fertőzött fájl méretének növekedését, hogy a fertőzést nehezebben lehessen észrevenni. A polimorf vírusok a terjedés során belső szerkezetüket állandóan változtatják, felismerésük így még nehezebb.

3. *Rombolás.* Miután a vírus saját számlálója, a belső óra állása, vagy valamilyen más jel alapján úgy dönt hogy eleget terjedt, felfedi kilétét. Vannak "jópofa" vírusok, amelyek zenélnek egyet, vagy valamilyen üzenetet írnak ki a képernyőre. A többség azonban rombol: tönkreteszi a háttértár fájljait, egyes vírusok a hardvert is visszavonhatatlanul károsítják.

A vírusok terjedése és az állami rendszerekben okozott károk következtében kialakultak a víruskereső- és irtó programok. Ezek a szoftverek képesek felismerni a vírusokat és a fertőzött állományokról többnyire el is tudják távolítani. Általában shareware programok. Az Interneten az egyre újabb és újabb vírusok keletkezése miatt kb. havonta jelenik meg a még több vírus detektálására alkalmas verzió. A vírusirtó programok közül magyar fejlesztők is komoly eredményeket értek el (Virusbuster, bővebb információ és vírusirtók letöltése: **`www.vbuster.hu`**).

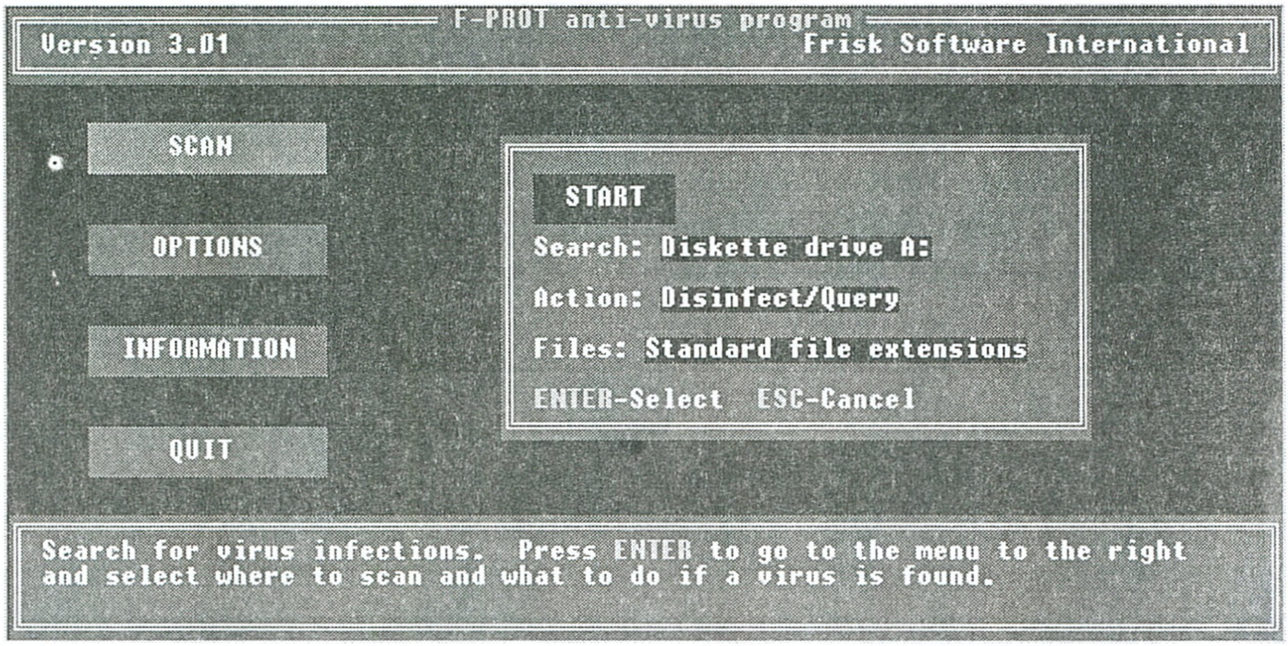
Több fajtájuk ismeretes:

- ***állandó vírusfigyelés:*** a program beépül az operációs rendszerbe és bármilyen fájlt vagy (hajlékonylemezen) boot szektort, amelyhez bármilyen célból fordulunk, előbb megvizsgál. Probléma esetén azonnal jelez.

- ***esetenkénti víruskeresés:*** a számítógépes programot a felhasználó indítja el pl. havonta (ekkor végigvizsgálja a teljes merevlemezt), vagy új program beszerzése esetén a floppyt vagy CD-t vizsgáltatja meg.

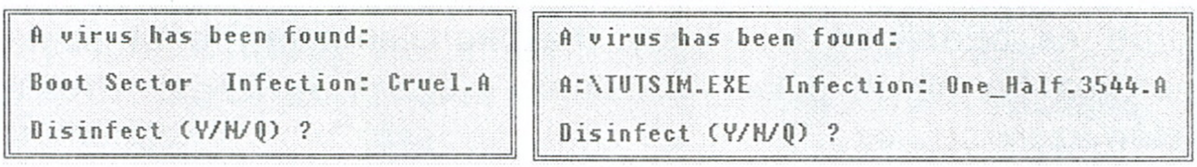
- ***hardveres vírusfigyelés:*** egyes számítógépekben hardveresen oldották meg a vírusok figyelését: a számítógép figyelmeztet, ha egy program valami szokatlant csinál (pl. a Master Boot Recordba ír).

Jelenleg nagyon sok jó víruskereső- és irtó program van. Az egyik közülük az izlandi fejlesztésű F-PROT:



Az F-PROT 1998. júniusi változata

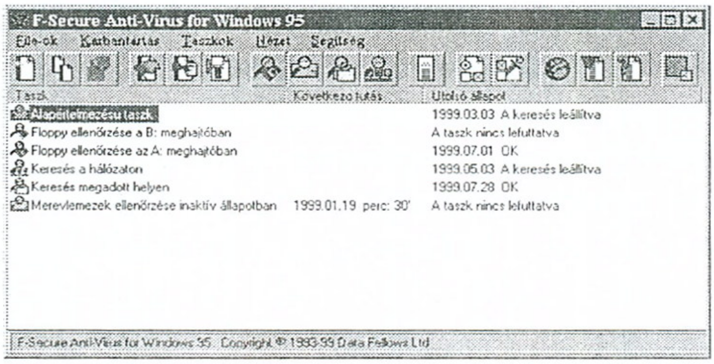
Vírusfertőzés esetén – ha tudja – leirtja a vírust:

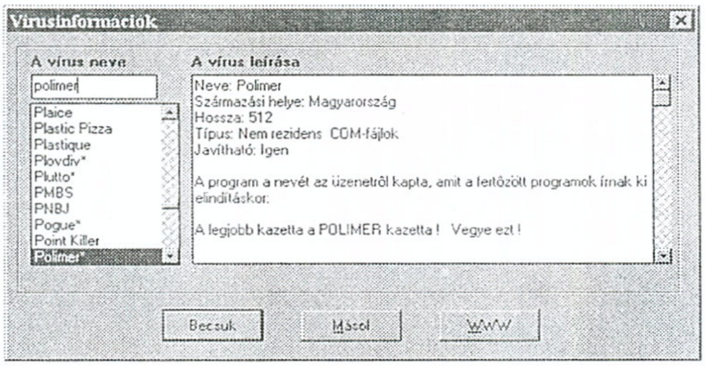


Az F-PROT boot- illetve fájl vírust talált

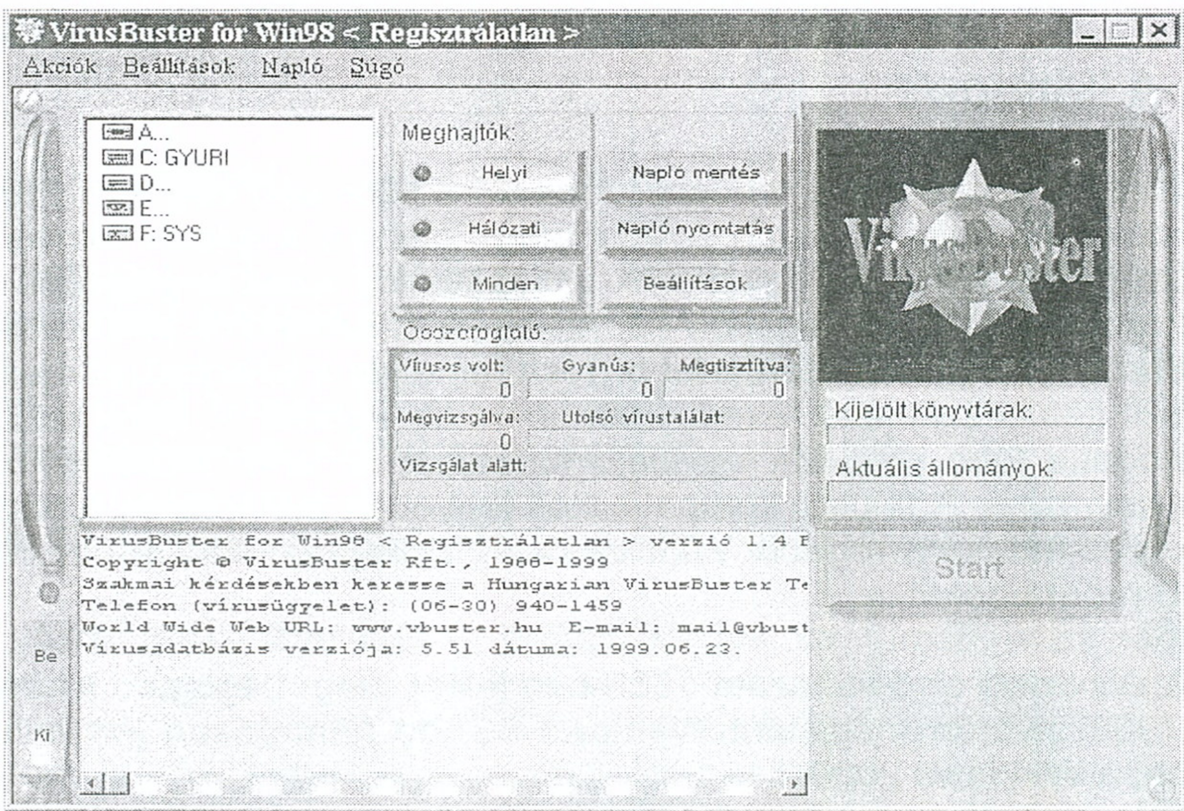
Ha nem tudja leirtani a vírust, a fertőzött fájlt le kell törölni. Nem irtható boot vírus esetén súlyosabb a probléma. Ha a winchesteren van boot vírus, indítsuk újra a gépet egy vírusmentes rendszerlemezről (ehhez esetleg a bootolási sorrendet át kell állítani a CMOS Setup-ban "**A: C:**"-ra) és utána indítsuk el a vírusirtót!

Az utóbbi években egy újfajta vírustípus jelent meg: a Word, illetve Excel fájlokban található ún. ***makróvírusok***. A programrendszer makró nyelvén írták őket és a Word illetve Excel dokumentumok segítségével terjednek, azaz nem csak IBM PC-n belül találhatók, hanem átterjedhetnek más platformokra is, pl. Apple Macintoshra.





Az F-Secure Anti-Virus for Windows 95 induló képernyője, valamint egy vírus leírása



A magyar fejlesztésű VirusBuster Windows 98 alatt futó változata

Jelenleg néhány tízezer vírus létezik. Ha időben felfedezzük egy vírus jelenlétét, ne essünk kétségbe, a jelenlegi programokkal többségük biztonságosan kiirtható. Ne felejtsük el rendszeresen frissíteni víruskereső programunkat (programjainkat) és ha az Internetről letöltünk egy fájlt, e-mailen küld valaki, esetleg CD-n vagy floppyn kapunk "ez egy tuti program, feltétlenül próbáld ki!", indítás előtt ne felejtsük el megvizsgálni. Ezen kívül *akárkitől* kapunk egy Word vagy Excel fájlt, először eresszünk rá egy makróvírus keresőt. Office 97 használata esetén globálisan letilthatjuk a makrók használatát – ez is segíthet. Rendszeres időközönként minden fontos anyagunkról (program/munka) készítsünk biztonsági másolatot!

A vírusokkal részletesen a negyedéves "Számítógépvírusok" tantárgy foglalkozik. Számos dologról nem esett szó, pl. a nagygépes rendszerekben található férgekről.

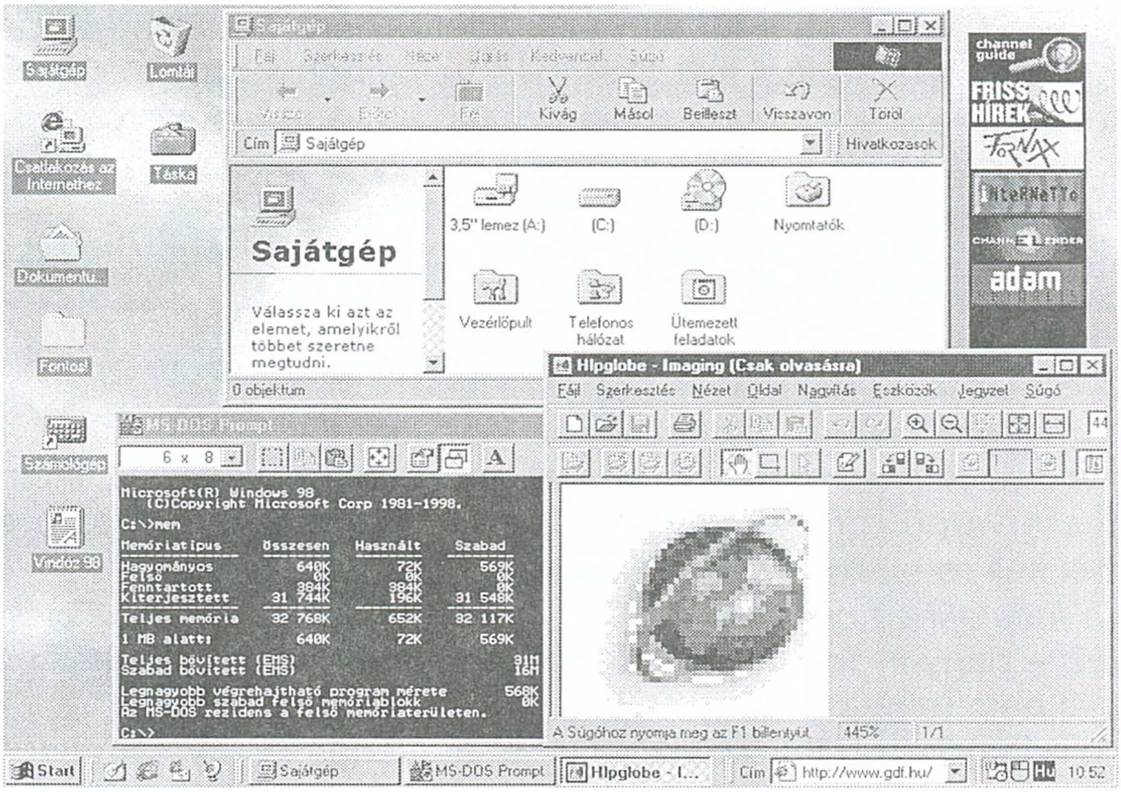
A vírusoknak nagy szakirodalma van. Egy ajánlható mű az alábbi:
- Tóth J. Szabolcs: PC vírusok, LSI Oktatóközpont Budapest 1999.

13. A Windows

Az IBM PC-k operációs rendszer "harcát" a Microsoft cég terméke, a Windows (angolul ablakok) nyerte meg, jelenleg a gépek túlnyomó többségén a Windows valamelyik változata fut, azaz lényegében világszabvány lett. A régebbi operációs rendszerekkel **ellentétben** ***grafikus felhasználói felület***et biztosít, amelyet a billentyűzet mellett egy mutató eszközzel – leggyakrabban az egérrel – kezelhetünk. Támogatja több program egyidejű futtatását is, amit ***multitasking****nak nevezünk.* A felülről kompatibilitás jegyében a Windows többnyire képes futtatni a régebbi Windows alá írt programokat, sőt az MS-DOS programokat is.

!

A Windows első változata 1985-ben jelent meg. Legújabb kiadása az 1998. nyarán megjelent Windows 98, de hamarosan jön a legújabb, a Windows 2000.



A Windows 98 képernyője

A Windowst és a különféle Windows alá készített alkalmazásokat részletesebben későbbi tantárgyakban beszéljük meg.

14. Az Internet

Az Internet egy világhálózatot kezelő szoftver (egy nagyon magasszintű „hálózati operációs rendszer").

Nézzük meg, hogy mire használhatjuk az Internetet, vagy ahogyan manapság nevezik, az "információs szupersztrádát":

14.1. Elektronikus levelezés (e-mail)

Az elektronikus levelezés – e-mail, kiejtve kb. "ímél" – az utóbbi években világszerte elterjedt és a különböző vállalatok, illetve nagyon sok ember a hagyományos postai elérés mellett rendelkezik e-mail címmel is. Az e-mail gyorsasága és olcsósága miatt erőteljesen megváltoztatta a telefonálási, illetve levélküldési szokásokat, aki tudott áttért az elektronikus kommunikációra. A gyorsaságra példa: egy

e-mail kb. 10 perc alatt jut el másik kontinensre; az olcsóságra: legfeljebb helyi telefondíjat kell fizetni és valamennyi havi előfizetési díjat. A telefonvonalon egy oldalnyi szöveg kb. 1 másodperc alatt kerül továbbításra. Az USA egyes vidékein a helyi beszélgetés díja benne foglaltatik a havi előfizetésben, így az otthonról történő e-mailezés (pl. Juno) és Web böngészés (pl. NetZero) lényegében teljesen ingyenes (csak az a „baj", hogy sok reklámot kapunk!).

Az e-mail lehetőségei:

- *szöveges levelet küldhetünk és fogadhatunk*. A küldéshez a címzett e-mail címét ismernünk kell (a cím felépítése jellemzően név@szervernév.domainnév), a levelek fogadásához saját e-mail címmel kell rendelkeznünk;
- *a levelekhez csatolva ("attachment") tetszőleges fájlokat is küldhetünk* – nagyon fontos lehetőség! (egyes ingyenes levelezőprogramoknál nem tehetjük meg);
- amennyiben "előfizetünk" – lehet ingyenes is – valamely *levelezési listára*, naponta kaphatunk onnan levelet, amely a kiválasztott téma híreit tartalmazza.

Az e-mail használatának két fontos alapszabálya:

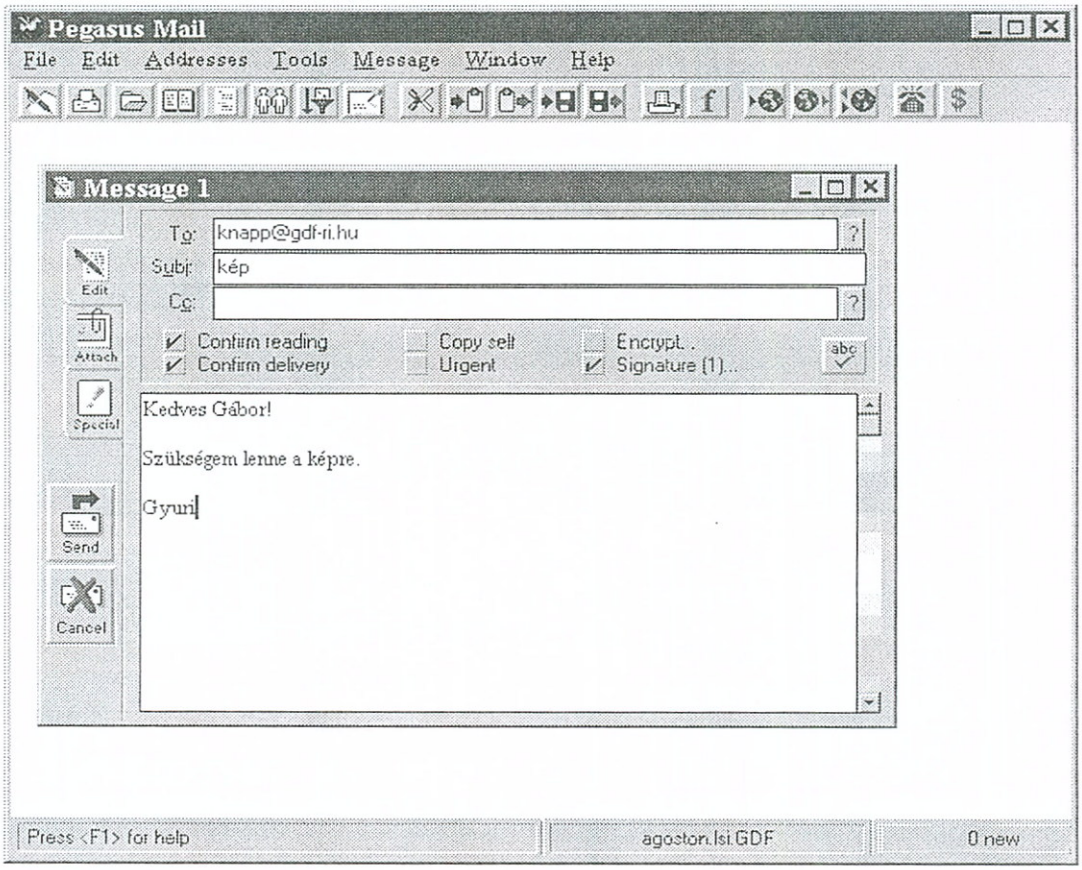
- nem biztos, hogy az elküldött levél megérkezik (többnyire azért megérkezik),
- nem biztos, hogy levelünket más nem olvassa el (az e-mail sok számítógépen halad keresztül, ha akarnak, bele tudnak tekinteni),
- gondoljunk arra is, hogy a címzett tudja-e, akarja-e fogadni azt, amit küldünk neki,
- a fentieken túli általános szabályokat az ún. Netiquette tartalmazza.

Az e-mail a világméretben egymással összekapcsolt számítógépek saját kábelhálózatán, illetve publikus vonalakon keresztül terjed. Otthonról történő Internetezés esetén nem építik ki a vezetéket minden lakásba a számítógéphez, hanem felhasználják a már létező telefonhálózatot.

A piacon nagyon sok **levelezőprogram** szerezhető be, kezelésük viszont nagyon hasonló. Az egyik ilyen program az ingyenes Pegasus Mail. Segítségével röviden tekintsük át lehetőségeinket:

• levél írása

A program indítása után a  gombra kattintva, vagy a File-New message menüsor választásával kezdhetünk új levél írásába:

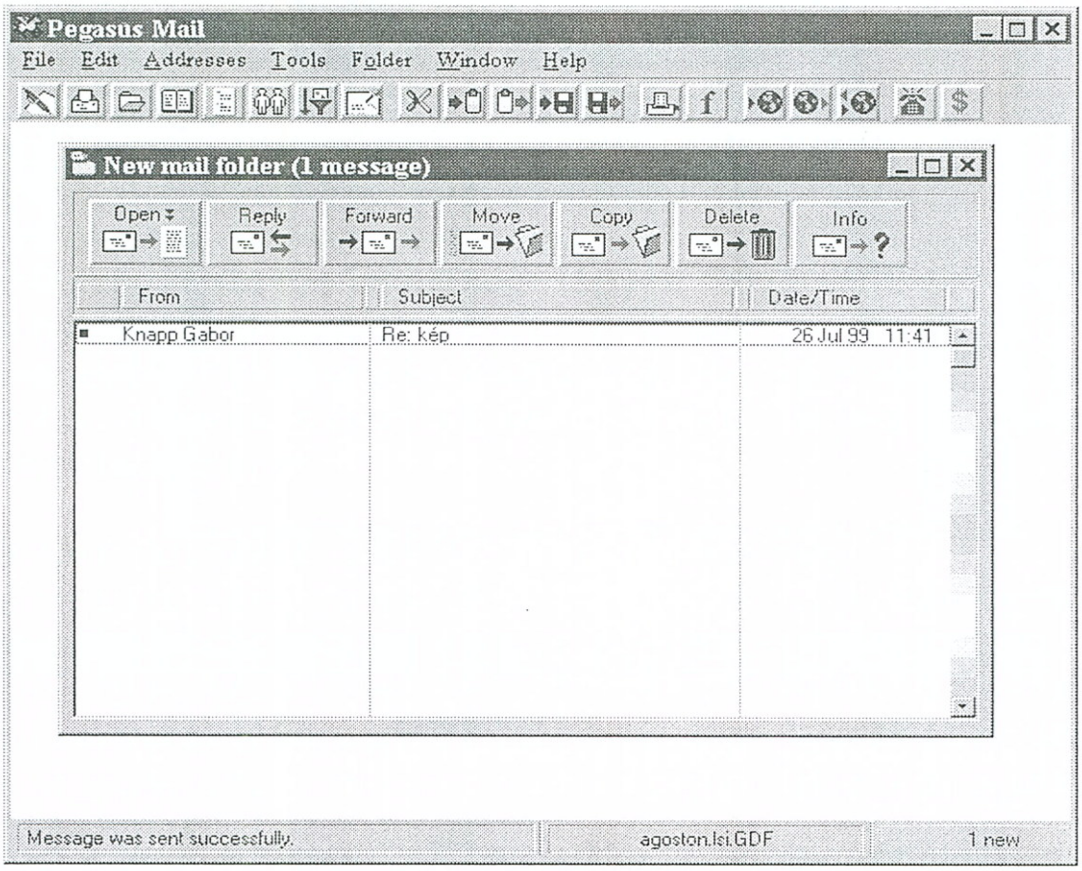


Levél írása

A megjelenő gyermekablakban a "To:" mezőbe írjuk a címzett e-mail címét. Ezt a mezőt pontosan kell kitöltenünk, különben a küldemény nem ér célba. A "Subj."-be (Subject) a levél címe vagy tárgya kerül, kitöltése nem kötelező. Amennyiben a levéllel fájl(oka)t is küldünk, az "Attach" gomb megnyomása után jelölhetjük ki a csatolandó állományokat. A levelet megírása után a "Send" gombbal küldhetjük el, a "Cancel"-lel visszalépünk a levélírásból. Ha bejelöljük a "Confirm delivery", illetve "Confirm reading" négyzeteket, a levél megérkezését, illetve elolvasását a címzett szerver számítógépe visszajelzi nekünk egy-egy e-mail formájában. Ez a lehetőség megtöbbszörözi a levélforgalmat és csak néhány szerver esetén működik.

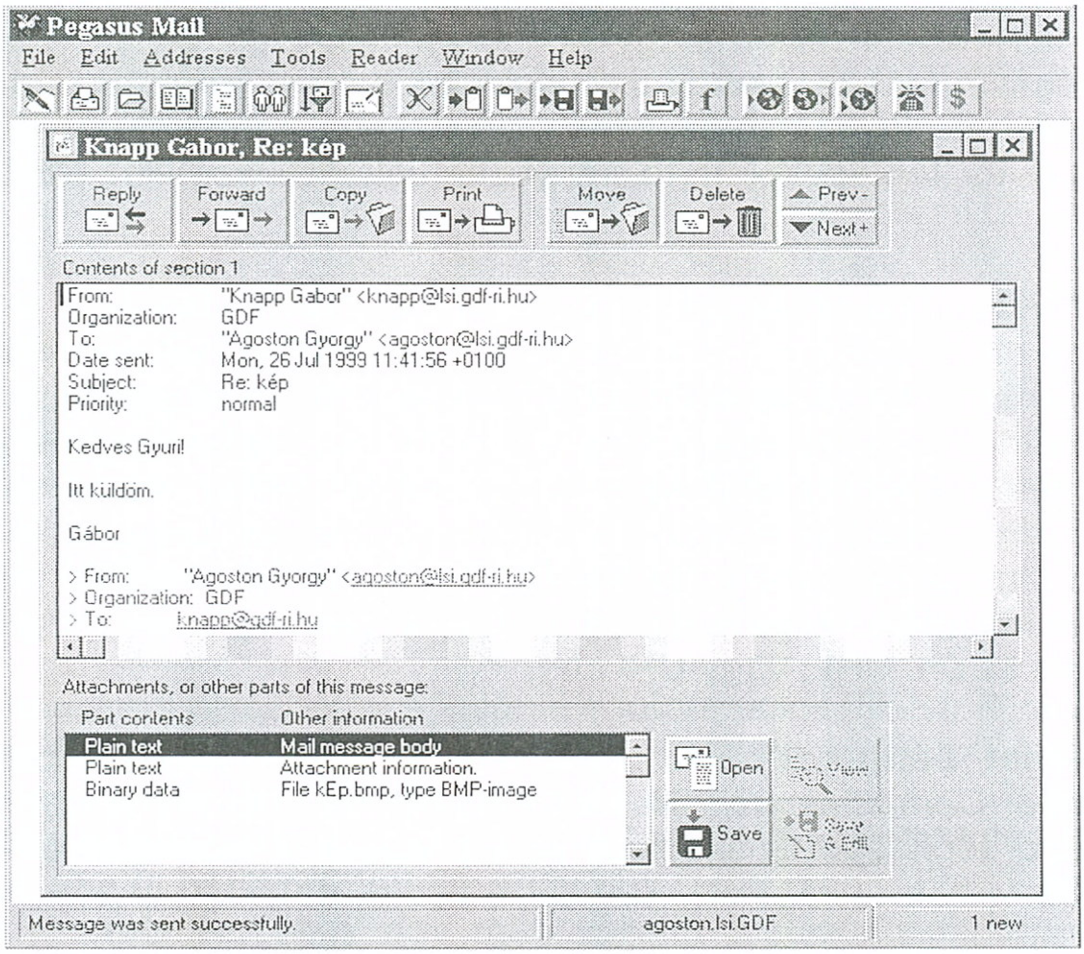
• **levél olvasása**

Érkezett leveleink a "New mail folder" gyermekablakban jelennek meg. Jelen példában egy levelet kaptunk. A From oszlopban látható a küldő neve, a Subject oszlopban a tárgya, a Date/Time oszlopban pedig az elküldés időpontja:



Levél olvasása 1.

A levél sorára kettőt kattintva olvashatjuk el tartalmát:



Levél olvasása 2.

A gyermekablak felső felében olvasható a levél szövege. Alul, az "Attachments" részből megállapíthatjuk, hogy egy kep.bmp nevű fájlt is csatoltak a levélhez. Erre a legalsó sorra kattintva, majd a Save gombot megnyomva a fájlt lementhetjük a háttértárra.

A levélre a Reply gomb megnyomása után válaszolhatunk. A Forward-dal továbbküldhetjük másoknak, a Print-tel kinyomtathatjuk, a Delete-tel pedig letörölhetjük.

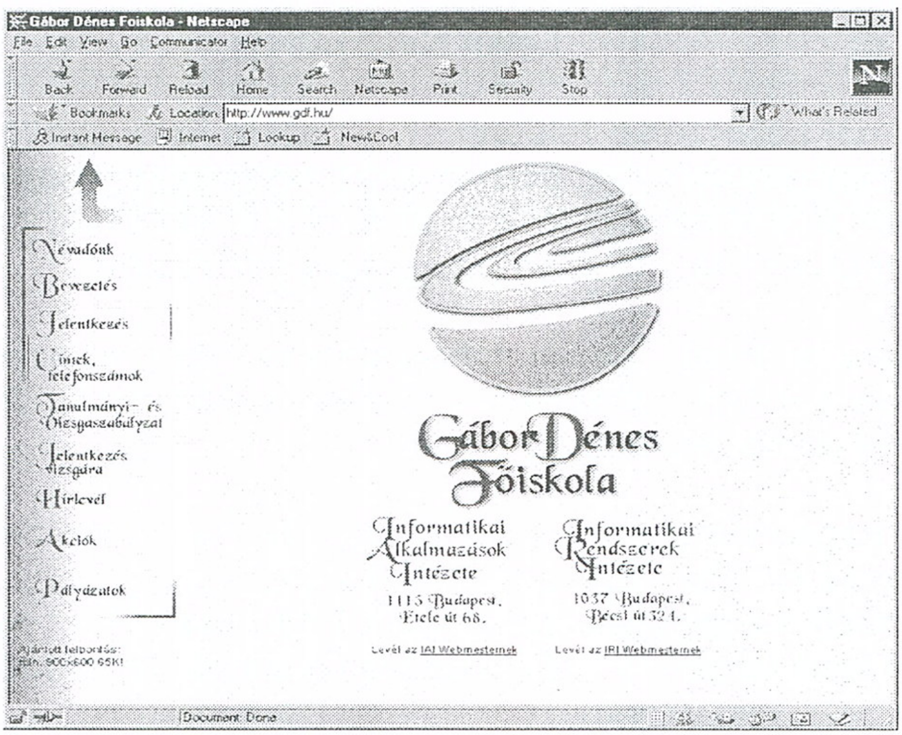
A fenti képen látható levelet a feladó a Reply – válaszadás – gomb megnyomásával írta. Ezt onnan vehetjük észre, hogy egyrészt a Subject mező megegyezik az általunk írttal, csak "Re:"-vel kezdődik, továbbá a levél végén ">" jellel kezdve olvasható az eredeti levél.

14.2. World Wide Web (WWW, "világméretű pókháló")

Az e-mail melletti másik leggyakoribb felhasználás. Az ún. "szörfözés" segítségével rengeteg információt tudhatunk meg: újságokat olvashatunk, archívumokban különféle adatokat kereshetünk, menetrendeket és térképeket tanulmányozhatunk és még kb. ezer más dolgot.

A WWW használatához – az e-mailhez hasonlóan – is tartozik egy-egy cím, csak ezt nem e-mail címnek hívják, hanem **weblap-**, vagy **honlap-cím**nek (eleinte az ottlap kifejezést is használták). Ezeken a címeken találhatók az egyes vállalatok, illetve magánszemélyek Internetre szánt közlendői. A levelezési címhez hasonlóan, ha valamit meg akarunk keresni, e címet is pontosan tudnunk kell, különben nem találjuk meg. Ha nem ismerjük, szerencsére felhasználhatjuk a különféle keresőprogramokat, amelyek segítenek megtalálni. Ez a cím az ún. URL, vagy URI cím.

Internet böngészőből is több létezik. A két legelterjedtebb a *Netscape Communicator*, illetve a *Microsoft Internet Explorer*. A Netscape segítségével tekintsük át a böngészők legfontosabb lehetőségeit:



A Netscape Communicator Internet web-böngésző

A *"Location"* mezőben található az aktuális, vagyis az éppen látható honlap címe, jelen esetben a GDF hivatalos honlapjának bejelentkező oldala (`www.gdf.hu`). Alatta, a nagy ablakban maga a honlap tartalma. A honlap tulajdonképpen az ún. HTML nyelven készült szöveges fájl, amelyet a böngésző értelmez és jelenít meg grafikusan. A fenti honlap valójában így néz ki (részlet):

```
<html>

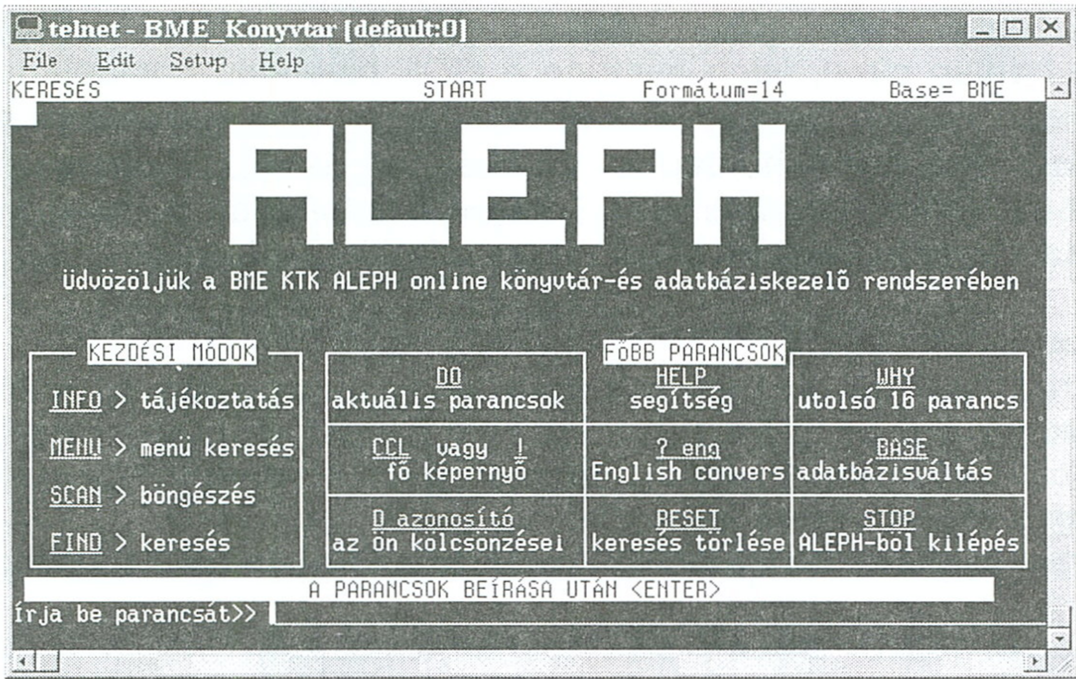
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=windows-1250">
<meta name="GENERATOR" content="Microsoft FrontPage
3.0">
<title>Gábor Dénes Foiskola</title>
</head>
```

A honlapon olvashatók a különféle információk, illetve itt találhatók az ún. ***link***ek (hivatkozások), melyek általában aláhúzott szövegek, vagy kis rajzok. Föléjük mozgatva az egeret, a kurzor alakja megváltozik, leggyakrabban egy mutató kézre. A linkre kattintva továbblépünk egy új honlapra, amely vagy egy "al-honlap", azaz valamelyik téma részletezve, vagy egy más vállalat bejelentkező képernyője, honlapja – lehet akár teljesen más világtájon is! Egyes honlapokon a hivatkozásra kattintva fájlokat tölthetünk le távoli szerverről – az Interneten hatalmas fájlgyűjtemények találhatók.

Az ablak felső részén található gombok jelentése: ha már több weblapot néztünk meg, a *Back*-kal és a *Forward*-dal lépkedhetünk hátra, illetve előre. A *Reload* újratölti az aktuális oldalt, a Home a böngésző indításakor bejelentkező oldalra tér vissza, a *Print*-tel nyomtatni tudunk, a *Stop*-pal pedig félbeszakíthatjuk az éppen folyó letöltést. A böngészők is rendelkeznek e-mail küldési és fogadási lehetőséggel.

14.3. Telnet

Segítségével egy ún. emulátorprogram saját számítógépünket egy távoli – többfelhasználós – számítógép termináljává alakítja át, így a távoli gép erőforrásait elérhetjük, programozhatjuk.

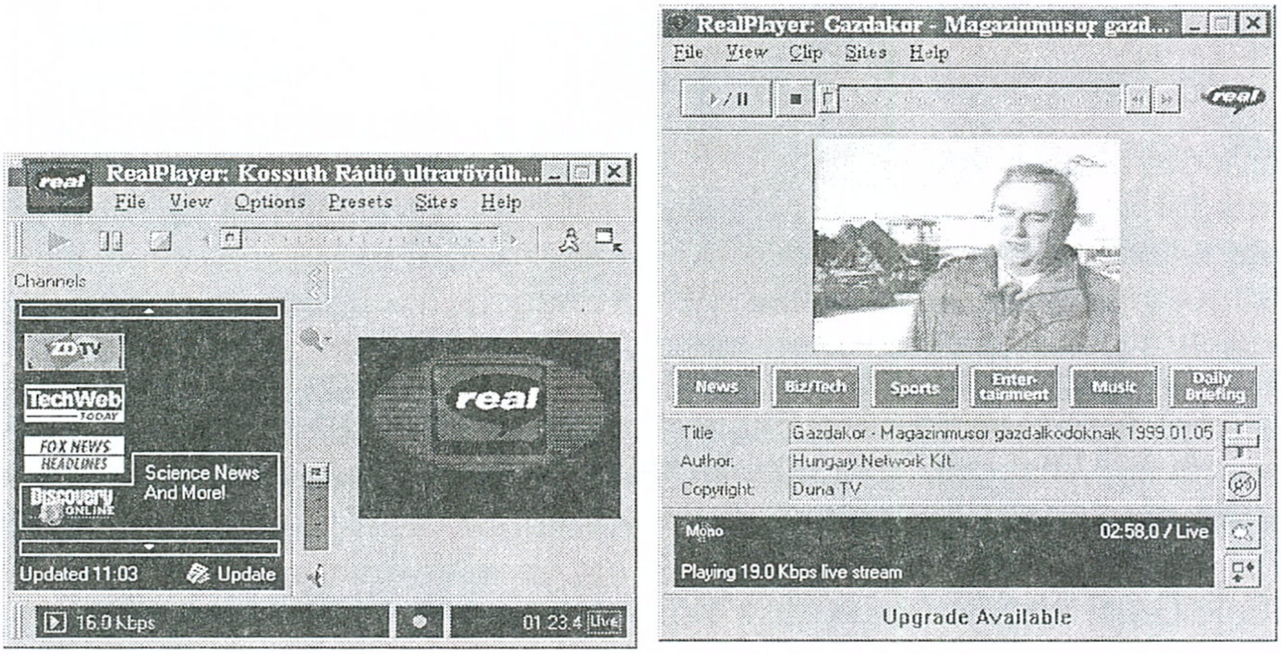


Telnet

A fenti képen a BME könyvtárának bejelentkező képe látható (`tulibb.kkt.bme.hu`). A Telnet kevéssé biztonságos volta miatt lassan kezdi felváltani egy újabb alkalmazás, az SSH.

14.4. Egyéb lehetőségek

A *RealPlayer* segítségével – a Windows 98 része – lehetőségünk nyílik **rádióadásokat** hallgatni, illetve **televízió műsorokat** nézni élőben, illetve az utóbbinál letöltés után fájlból. A jelenlegi csekély sávszélesség miatt a TV adás sajnos csak néhány képkocka másodpercenként és a képminőség is gyenge, de e lehetőségek főleg olyan földrajzi helyeken használhatók ki jól, ahol nem fogható az adás antennával. Pl. az USA-ban kitűnő minőségben hallgatható a Kossuth rádió – Interneten keresztül. Bővebb információ: `www.radio.hu`, `www.dunatv.hu`.



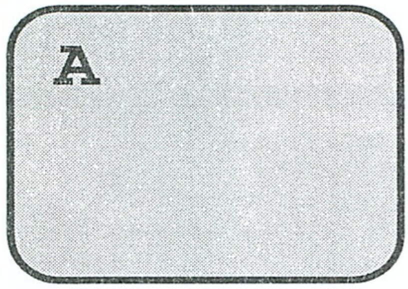
A RealPlayer segítségével rádió- és televízió műsorokat foghatunk

15. Néhány gondolat a programozásról

Vajon hogyan fogalmazhatjuk meg a számítógép működését röviden? A lényeg tömören: *bekapcsolástól kikapcsolásig nem történik más, mint a mikroprocesszor (CPU) egyfolytában utasításokat hajt végre, amelyeket számára előírunk*. Ezek az utasítások számokból állnak és "gépi kód"-nak nevezzük. Ebben különbözik a számítógép más berendezésektől (pl. üzenetrögzítős telefon, mosógép stb.), melyek bár tartalmaznak különféle chip-eket, sőt programokat, amelyek közül a felhasználó választhat, e programokat viszont nem tudjuk sem megváltoztatni, sem pedig mással helyettesíteni. !

! A programozás fontos jellemzője az *elágazás: a program adott pontjain külső (azaz a felhasználó által befolyásolható), vagy belső jellemzők értékétől függően a futás más-más helyen folytatódhat.*

Nézzük meg a legelterjedtebb személyi számítógéptípus, az IBM PC példáján keresztül a programozás fogalmát. Készítsünk el egy egyszerű feladatot: írassunk ki a képernyőre egy "A" betűt:



A felhasznált számítógép tehát egy IBM PC, vagy vele kompatíbilis gép. A programozás gyökeresen függ a használt operációs rendszertől is. Válasszuk a legelterjedtebbet: legyen az MS-DOS, illetve a Windows 95/98 DOS ablaka. A feladat megoldásai:

15.1. Gépi kód

A gépi kód a következő nyolc számból áll:

```
180, 2, 178, 65, 205, 33, 205, 32
```

Szokás hexadecimálisan is írni: B4h, 02h, B2h, 41h, CDh, 21h, CDh, 20h. Hogy miért éppen ezek a számok szükségesek, nehéz lenne elmagyarázni, elégedjünk meg annyival, hogy a CPU gépkönyvéből kikereshetők. Ha e számokat beírjuk egy fájlba és a fájlt elindítjuk (lefuttatjuk), a képernyőn megjelenik az "A" betű.

Azt azért már most észrevehetjük, hogy a gépi kódot (és csak ezt) a CPU tökéletesen megérti, de az ember lényegében egyáltalán nem. Ezért a programokat nem gépi kódban írják – bár megtehető -, hanem valamely programozási nyelven.

15.2. Assembly

Az Assembly – kiejtve kb. "eszembli" – programozási nyelven írt megoldás, az ún. forrásprogram az alábbi:

```
title A betu kiiro program Assemblyben
.model tiny
.code
org 100h
start:
 mov ah,2
 mov dl,'A'
 int 21h
```

```
int 20h
end start
```

Minden programozási nyelvre jellemző, *hogy egy feladat többféleképpen is elkészíthető*, a fenti néhány sor csupán az egyik lehetséges megoldás. Megértéséhez meg kell tanulni a nyelvet, hasonlóan ahhoz, mintha egy élő idegen nyelvet tanulnánk meg. Az Assemblyt tehát az ember megérti, a CPU viszont nem. Ezért a fenti programot egy *fordítóprogram, az ún. compiler – kiejtve kb. "kompájler" – átalakítja gépi kódra, amely már betölthető és az ún. linkelés (szerkesztés) után végrehajtható.* Az első compilert természetesen gépi kódban kellett megírni egyszer régen, mert amíg nincs, nem használható az Assembly és semmilyen más programozási nyelv.

A gépi kód és az Assembly között szoros kapcsolat van, emiatt a lefordított program pontosan abból a nyolc számból (8 bájtból) áll, amelyet az előző pontban már megismerhettünk! A szoros kapcsolat – gépközeliség – miatt *szokás az Assemblyt alacsony szintű programozási nyelv*nek nevezni. Ez a nyolc bájt természetesen csak az IBM PC-kben használt Intel CPU-kon fut jól, más mikroprocesszoroknál teljesen más utasításokat jelentenek.

A Debug segédprogram Assemblyről fordít gépi kódra és vissza. Segítségével láthatjuk, hogy a nyolc bájtnak mely Assembly utasítások felelnek meg (mindegyik szám hexadecimális; a 41h az "A" betű ASCII kódja):

```
-u
1CE0:0100 B402 MOV AH,02
1CE0:0102 B241 MOV DL,41
1CE0:0104 CD21 INT 21
1CE0:0106 CD20 INT 20
-
```

15.3. C nyelv

A C nyelvű forrásprogram az alábbi:

```
/* A betu kiiro program C-ben */
#include <stdio.h>
int main(void)
{
```

```
  printf("A");
  return 0;
}
```

A C nyelv nem Assembly, más a szintaxisa és mások az utasítások. Ehhez is tartozik egy compiler, amely gépi kódra fordítja a forrásprogram sorait. Ezt a compilert vagy gépi kódban készítették el, vagy a már használható Assemblyben.

! ***A C nyelv magas szintű programozási nyelv,*** *azok közül a legalacsonyabb, leginkább gépközeli, ezért rendszerprogramozási nyelvnek is nevezik.* A magas szintű nyelvek jellemzője, hogy a forrásprogram érthető angol szavakat tartalmaz. A történelem folyamán úgy alakult, hogy a számítástechnika nyelve nem a magyar lett, hanem az angol. Emiatt sokkal könnyebb dolgunk lesz, ha ismerjük egy kicsit. Amennyiben idegenkedünk az angoltól, mintegy száz speciális szó megtanulása után nagyjából kezelni tudunk minden angolul beszélő programot.

A keletkezett gépi kódú program viszont nem 8 bájtból fog állni, hanem (fordítótól függően) néhány ezer bájtból! Ennek oka, hogy a fordító különféle ellenőrző rutinokat, copyright szövegeket és egyéb függvényeket is hozzárak a programhoz, ezt a funkcióját ki sem tudjuk iktatni. Ráadásul emiatt a program lassabban is fut, mint az Assembly változata! Azaz ***Assemblyben készíthető el a leggyorsabban futó és legrövidebb méretű program***. A magas szintű programnyelveknek van még egy hátránya: nem tartalmazzák a számítógép kihasználásának összes lehetőségét, amelyet gépi kódban és Assemblyben megvalósíthatunk. Legnagyobb előnyük viszont az érthetőség.

!

A C nyelv és továbbfejlesztett változata a C++ nagyon elterjedt, a Windows alkalmazások többségét ebben készítik. A C-nek létezik szabványosított változata (ANSI C) és gyakorlatilag minden számítógép típushoz létezik C fordító, azaz forrásprogram szinten a C-ben írt programok hordozhatók, idegen szóval ***platformfüggetlenek***. Az Assemblyt olyan speciális helyeken használják, ahol a gyorsaság, a kis méret és a számítógép összes lehetőségének elérése számít (operációs rendszerek, vezérlőprogramok). A Windowst Assembly és C nyelven írták.

15.4. Pascal

Végül pedig következzen a megoldás Turbo Pascalban:

```
{ A betu kiiro program Pascalban }
program abetu;
begin
 write('A')
end.
```

A *Pascal nyelv* egy újabb programozási nyelv, saját szintaxissal és utasításokkal. Magas szintű programozási nyelv. A lefordított program a C-hez hasonlóan néhány ezer bájtból áll és természetesen lassabb, mint az Assembly változat.

A Pascal kevésbé elterjedt, mint a C. Leginkább az oktatásban használják, mert az egyik legkönnyebben elsajátítható nyelv. Továbbfejlesztett változata az egyre nagyobb népszerűségnek örvendő Delphi, mely Windows alkalmazások készítésére használható.

A fenti példákból látható, hogy egy-egy feladat több programozási nyelven is elkészíthető. A választást a feladat jellege és egyéb külső tényezők határozzák meg.

Egy új programozási nyelv tanulásakor az első feladat általában a "Helló világ!" szöveg kiíratása a képernyőre. A `http://www.latech.edu/~acm/HelloWorld.shtml` weblapon számtalan nyelven megtalálható e feladat megoldása.

15.5. A programkészítés lépései

Az alábbiakban tekintsük át egy kisebb feladat megoldásának lépéseit. A leírt sorrend egy javaslat a sok közül, másféle sorrend és lépések is elképzelhetők.

- Feladat pontos megfogalmazása (input, output adatok). Ha már itt elrontjuk, sok gondot okozhat a későbbiekben, lehet hogy elölről kell kezdenünk az egészet! !
- Feladat + tudás + egyéb igények alapján hardver (gép) és szoftver (operációs rendszer és programozási nyelv) kiválasztása.

- A probléma *számítógépes jellegű* megfogalmazása (pl. rasztergrafika → pontokból áll → matematikailag egymás után milyen számokat vegyen fel a rajzoló függvény)
- A nagy feladat részekre bontása
- Az egyes részek megírása (programozás, forrásprogram, egy ember vagy team), tesztelés
- A részek összeillesztése
- Fordítás, a teljes program előállítása (futtatható állapot)
- Tesztelés, hibakeresés és javítás (minden menüpontot, összes választási lehetőséget, extrém esetek, felhasználóbiztos legyen)
- Végső szépítés (a felhasználó igénye szerint)
- Védelem beépítése (másolás, módosítás, használati időtúllépés esetére)
- Tesztelés, hibakeresés és javítás
- Dokumentáció elkészítése (Felhasználói kézikönyv)
- Telepítés (installálás)
- Betanítás azoknak, akik használni fogják a programot
- Szervizelés, karbantartás, garancia
- Továbbfejlesztés

Ha bárhol problémába ütközünk, egy korábbi lépéshez kell visszatérnünk. Szélsőséges esetben akár az első ponthoz, amely így megváltoztatja az egész munkafolyamatot. Ezért fontos az első lépés kellő átgondolása.

! A fenti felsoroláshoz még egy nagyon fontos szempont tartozik: *gazdasági számítások*: mennyibe kerül a fejlesztés, megéri-e fejleszteni vagy inkább a piacon már létező terméket vásároljuk meg, ki fog fizetni, piac, marketing stb.

Egy adott programnyelv elsajátítása természetesen több időt vesz igénybe. A fentieken csupán egy kis ízelítőt adtunk a programozás világából.

? Ellenőrző kérdések

1. Mit jelentenek a következő kifejezések: hardver, szoftver, shareware, freeware, felülről kompatíbilis, fájl, bootolás, bit, bájt, kilobit, kilobájt, Baud, ROM, RAM, UPS, CMOS, Y2K, BSA?
2. Egy mai számítógép milyen hardver és szoftver részekből tevődik össze?
3. Hány adatvezetéke van a soros és a párhuzamos portnak?

4. Mit jelent a SCSI és az USB?
5. Mekkora a BIOS által elvileg kezelhető merev- és hajlékonylemez maximális mérete?
6. Hogyan működik a mátrix-, a tintasugaras, a lézer- és a hőnyomtató?
7. Hogyan lehet DOS-ból és a Commander program segítségével könyvtárat átnevezni, illetve törölni?
8. Mire használhatók a batch fájlok?
9. Készítsen batch fájlt, mely az `autoexec.bat`-ban átállítja a promptot úgy, hogy a parancssor felett mindig megjelenik az *"Éljenek a kék lepkék!"* üzenet!
10. Mire való a formattálás és hogyan hajtható végre?
11. Mit jelent az alacsony szintű formázás?
12. Mit jelent a meleg- és hidegindítás, hogyan hajtható végre?
13. Hogyan hajtható végre egy Commander program segítségével több fájl másolása floppyra?
14. Egy Word dokumentum kb. hányad részére tömöríthető össze?
15. Mit jelent az "önkicsomagoló archívum" és hogyan készíthető?
16. Hogyan szerezhetünk vírust?
17. Mi a makróvírus?
18. Ismertesse a Windows legfontosabb jellemzőit!
19. Mire használható az e-mail?
20. Mit jelent az e-mail cím és az attachment?
21. Mi kerül a levelezőprogramnál a Subject mezőbe?
22. Mit jelent a honlap, a link és az URL?
23. Mi történik, ha egy böngészőben a Home, illetve a Reload gombra kattintunk?
24. Ismertesse a Telnet fogalmát és jellemzőit!
25. Ismertesse a RealPlayer fogalmát és jellemzőit!
26. Melyek a gépi kód és az Assembly tulajdonságai és kapcsolata?
27. Mi a C nyelv jellemzője?
28. Mit jelent az, hogy a C nyelv platformfüggetlen?
29. Ismertesse egy egyszerűbb programozási feladat megoldásának lépéseit!

IRODALOMJEGYZÉK

1. Aspinall, D.: The Microprocessor and 1st Application. ISBN 0-421-22241-9
2. Andrew, M.: Programming Microprocessor Interfaces for Control and Instrumentation. ISBN 0-13-729996-6
3. Burton, D. P. – Dexter, A.: Microprocessor System Handbook. Analog Devices Inc., USA, 1983
4. Artwick, B.A.: Microcomputers Interfacing. Prentice Hall USA, 1990
5. Coffron, J.W.: Microprocessors Maintance and Repair. ISBN 0-13-694273-3
6. Cole, J.: Pocket Computer Programming Made Easy. ISBN 0-86668-009-8
7. Darcy, L. – Boston, L.: Dictionary of Computer Terms. Simon and Schuster Inc., Clevland, USA, 1988
8. Evans, C.: The Mighty Micro. London, Kogan Page ISBN 0-575-027088
9. Hilbrun, L. – Julish, P. M.: Microcomputers, Microprocessors Elektronics Book Series. McGraw-Hill Co., London, 1991
10. Hornby, A. A.: Oxford Advanced Learner's Dictionary. Oxford University Press, 1989. ISBN 0-19-431136-8
11. Kovács Magda: Mikroelektronika – mikroszámítógép értelmező szótár. LSI. Budapest, 1993
12. Kovács Magda: Egyszerűen a mikroszámítógépekről. LSI. Budapest, 1993
13. Klingmann, E. E.: Microprocessor Systems Design. ISBN 0-13-581413-8
14. Meadows, A. I. – Gordon, M. – Singleton, A.: Dictionary of New Information Technology. Kogan Page, London, 1989
15. Ogdin, C. A.: Software Design for Microprocessors. ISBN 0-1382-1744-0
16. Oleksy, J. E.: Microprocessors and Digital Computer Technique. ISBN 0-13-581116-3
17. Osborne, A.: An Introduction to Microcomputers. Sybex, USA, 1994
18. Osborne, A.: 80386/80286 Assembly Language Programming. McGraw-Hill, USA, 1986
19. Nelson, R. P.: Microsoft's 80386/80486 Programming Guide. USA, 1991
20. Simonyi Károly: A fizika kultúrtörténete. Gondolat Kiadó, Budapest, 1978

21. Sippl, C. I.: Microcomputer Dictionary. Howard W. Sams Co. Inc., Indiana, USA, 1992
22. Sloan, M. E.: Introduction to Mini and Microcomputers. ISBN 0-201-07290-4
23. De Voney, Ch. – Summe, R.: IBM's Personal Computer. USA, 1988
24. Whitbread, M.: Microprocessor Software. ISBN 0-7194-0013-9
25. Murray, W. H. – Pappas, Ch. H.: A 80386/80286-os processzor Assembly nyelvű programozása. Budapest, 1988
26. Willis, J. – Miller, M.: Computer for Everybody. ISBN 0-918398-49-5
27. Whitlock, Q. – Dean, Ch.: A Handbook of Computer Based Training. Kogan Page, London, ISBN 0-85038-557-1
28. Wood, A.: Microprocessors. ISBN 0-03-910313-7
29. Zaks, R. – Lesea, A.: Microprocessor Interfacing Technicues ISBN 0-89588-029-6
30. Intel 486™ DX2 Microprocessor Data Book. USA, 1992
31. Pentium™ Processor User's Manual Volume 3; Architecture and Programming Manual. Intel, USA, 1993
32. Sághi Balázs: Alaplapok, sínrendszerek, konfigurálás. Panem-McGraw-Hill, Budapest 1996
33. Ila László – Sághi Balázs: Megjelenítők, háttértárolók, soros és párhuzamos interfész. Panem-McGraw-Hill, Budapest 1996
34. D. E. Knuth: A számítógép programozás művészete 1., 2., 3. kötet. Műszaki Könyvkiadó, Budapest 1997
35. Peter Norton: Az IBM PC programozása (4. kiadás). Műszaki Könyvkiadó 1996
36. Arthur Dickschus: Egyszerűen PC ismeretek. Hardver. Panem-McGraw-Hill, Budapest 1998
37. Arthur Dickschus: Egyszerűen PC ismeretek. Szoftver. Panem-McGraw-Hill, Budapest 1998
38. Andrew S. Tanenbaum Albert S. Woodhull: Operációs rendszerek. Panem-McGraw-Hill, Budapest 1999
39. Knapp Gábor: Operációs rendszerek. LSI. Budapest 1999
40. Kovács Magda: Első lépés a mikroszámítógépek világába. LSI. Budapest, 1996.

MELLÉKLET

ASCII kódtábla

DECIMÁLIS			0	16	32	48	64	80	96	112
	HEXADECIMÁLIS		00	10	20	30	40	50	60	70
		BINÁRIS	0000 0000	0001 0000	0010 0000	0011 0000	0100 0000	0101 0000	0110 0000	0111 0000
0	0	0000	NUL	DLE	SPACE	0	@	P	`	p
1	1	0001	SOH	DC1	!	1	A	Q	a	q
2	2	0010	STX	DC2	"	2	B	R	b	r
3	3	0011	ETX	DC3	#	3	C	S	c	s
4	4	0100	EOT	DC4	$	4	D	T	d	t
5	5	0101	ENQ	NAK	%	5	E	U	e	u
6	6	0110	ACK	SYN	&	6	F	V	f	v
7	7	0111	BEL	ETB	'	7	G	W	g	w
8	8	1000	BS	CAN	(	8	H	X	h	x
9	9	1001	HT	EM	)	9	I	Y	i	y
10	A	1010	LF	SUB	*	:	J	Z	j	z
11	B	1011	VT	ESC	+	;	K	[	k	{
12	C	1100	FF	FS	,	<	L	\	l	\|
13	D	1101	CR	GS	-	=	M	]	m	}
14	E	1110	SO	RS	.	>	N	^	n	~
15	F	1111	SI	US	/	?	O	_	o	DEL

Például az '6' karakter kódja (függőleges + vízszintes)

decimális:	48 + 6 =	54
hexadecimális:	30 + 6 =	36
bináris:	00110000 + 0110 =	00110110

ANSI 1250 (Magyar Windows 3.1/95 – Magyar szabvány)

DECIMÁLIS			128	144	160	176	192	208	224	240
	HEXADECIMÁLIS		80	90	A0	B0	C0	D0	E0	F0
		BINÁRIS	1000 0000	1001 0000	1010 0000	1011 0000	1100 0000	1101 0000	1110 0000	1111 0000
0	0	0000	□	□		°	Ŕ	Đ	ŕ	đ
1	1	0001	□	‘	ˇ	±	Á	Ń	á	ń
2	2	0010	‚	’	˘	˛	Â	Ň	â	ň
3	3	0011	□	_	Ł	ł	Ă	Ó	ă	ó
4	4	0100	„	•	¤	´	Ä	Ô	ä	ô
5	5	0101	…	–	Ą	µ	Ĺ	Ő	ĺ	ő
6	6	0110	†	—	¦	¶	Ć	Ö	ć	ö
7	7	0111	‡	□	§	·	Ç	×	ç	÷
8	8	1000	□	™	¨	¸	Č	Ř	č	ř
9	9	1001	‰	š	©	ą	É	Ů	é	ů
10	A	1010	Š	›	Ş	ş	Ę	Ú	ę	ú
11	B	1011	‹	ś	«	»	Ë	Ű	ë	ű
12	C	1100	Ś	ť	¬	Ľ	Ě	Ü	ě	ü
13	D	1101	Ť	ž	-	˝	Í	Ý	í	ý
14	E	1110	Ž	ź	®	ľ	Î	Ţ	î	ţ
15	F	1111	Ź	ź	Ż	ż	Ď	ß	ď	˙

ANSI 1252 (Amerikai Windows 3.1/95)

DECIMÁLIS			128	144	160	176	192	208	224	240
	HEXADECIMÁLIS		80	90	A0	B0	C0	D0	E0	F0
		BINÁRIS	1000 0000	1001 0000	1010 0000	1011 0000	1100 0000	1101 0000	1110 0000	1111 0000
0	0	0000	□	□		°	À	Đ	à	đ
1	1	0001	□	‘	¡	±	Á	Ñ	á	ń
2	2	0010	‚	’	¢	²	Â	Ò	â	ň
3	3	0011	ƒ	_	£	³	Ã	Ó	ã	ó
4	4	0100	„	•	¤	´	Ä	Ô	ä	ô
5	5	0101	…	–	¥	µ	Å	Õ	å	ő
6	6	0110	†	—	¦	¶	Æ	Ö	æ	ö
7	7	0111	‡	˜	§	·	Ç	×	ç	÷
8	8	1000	ˆ	™	¨	¸	È	Ø	è	ř
9	9	1001	‰	š	©	¹	É	Ù	é	ů
10	A	1010	Š	›	ª	º	Ê	Ú	ê	ú
11	B	1011	‹	œ	«	»	Ë	Û	ë	ű
12	C	1100	Œ	□	¬	¼	Ì	Ü	ì	ü
13	D	1101	□	□	-	½	Í	Ý	í	ý
14	E	1110	□	Ÿ	®	¾	Î	Þ	î	ţ
15	F	1111	□	Ÿ	¯	¿	Ï	ß	ï	˙

ASCII 852 (DOS – Latin 2 kiterjesztés, Magyar szabvány)

DECIMÁLIS			128	144	160	176	192	208	224	240
	HEXADECIMÁLIS		80	90	A0	B0	C0	D0	E0	F0
		BINÁRIS	1000 0000	1001 0000	1010 0000	1011 0000	1100 0000	1101 0000	1110 0000	1111 0000
0	0	0000	Ç	É	á	□	‹	đ	Ó	-
1	1	0001	ü	Ĺ	í	□	□	Đ	ß	˝
2	2	0010	é	ĺ	ó	,	‘	Ď	Ô	˛
3	3	0011	â	ô	ú	□	’	Ë	ń	ˇ
4	4	0100	ä	ö	Ą	„	“	ď	ń	˘
5	5	0101	ů	Ľ	ą	Á	”	Ň	ň	§
6	6	0110	ć	ľ	Ž	Â	Í	Š	÷	A
7	7	0111	ç	Ś	ž	Ě	ă	Î	š	¸
8	8	1000	ł	ś	Ę	Ş	•	ě	Ŕ	°
9	9	1001	ë	Ö	ę	…	–	©	Ú	¨
10	A	1010	Ő	Ü	¬	†	—	®	ŕ	˙
11	B	1011	ő	Ť	ź	‡	□	±	Ű	ű
12	C	1100	î	ť	Č	□	™	µ	ý	Ř
13	D	1101	Ź	Ł	ş	Ż	›	Ţ	Ý	ř
14	E	1110	Ä	×	«	ż	¦	Ů	ţ	·
15	F	1111	Ć	č	»	‰	¤	¶	´	

ASCII 437 (DOS – PC8 kiterjesztés)

DECIMÁLIS			128	144	160	176	192	208	224	240
	HEXADECIMÁLIS		80	90	A0	B0	C0	D0	E0	F0
		BINÁRIS	1000 0000	1001 0000	1010 0000	1011 0000	1100 0000	1101 0000	1110 0000	1111 0000
0	0	0000	Ç	É	á	░	└	╨	α	≡
1	1	0001	ü	æ	í	▒	┴	╤	ß	±
2	2	0010	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	0011	â	ô	ú	│	├	╙	π	≤
4	4	0100	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	0101	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	0110	å	û	ª	╢	╞	╓	µ	÷
7	7	0111	ç	ù	º	╖	╟	╫	τ	≈
8	8	1000	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	1001	ë	Ö	⌐	╣	╔	┘	Θ	·
10	A	1010	è	Ü	¬	║	╩	┌	Ω	·
11	B	1011	ï	¢	½	╗	╦	█	δ	√
12	C	1100	î	£	¼	╝	╠	▄	∞	^{n}
13	D	1101	ì	¥	¡	╜	═	▌	φ	^{2}
14	E	1110	Ä	₧	«	╛	╬	▐	ε	■
15	F	1111	Å	ƒ	»	┐	╧	▀	∩	

ASCII CWI (DOS)

DECIMÁLIS			128	144	160	176	192	208	224	240
	HEXADECIMÁLIS		80	90	A0	B0	C0	D0	E0	F0
		BINÁRIS	1000 0000	1001 0000	1010 0000	1011 0000	1100 0000	1101 0000	1110 0000	1111 0000
0	0	0000	Ç	É	á	░	└	╨	α	≡
1	1	0001	ü	æ	í	▒	┴	╤	ß	±
2	2	0010	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	0011	â	ő	ú	│	├	╙	π	≤
4	4	0100	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	0101	à	Ó	Ñ	╡	┼	╒	σ	⌡
6	6	0110	å	ű	ª	╢	╞	╓	µ	÷
7	7	0111	ç	ù	Ő	╖	╟	╫	τ	≈
8	8	1000	ê	Ű	¿	╕	╚	╪	Φ	°
9	9	1001	ë	Ö	⌐	╣	╔	┘	Θ	∙
10	A	1010	è	Ü	¬	║	╩	┌	Ω	·
11	B	1011	ï	¢	½	╗	╦	█	δ	√
12	C	1100	Í	£	¼	╝	╠	▄	∞	^{n}
13	D	1101	ì	¥	¡	╜	═	▌	φ	^{2}
14	E	1110	Ä	₧	«	╛	╬	▐	ε	■
15	F	1111	Á	ƒ	»	┐	╧	▀	∩	

Változások a 437-es kódlaphoz képest: Á(143), Í(140), Ó(149), ő (147), Ő(167), ű(150), Ű(152)

Magyarosított kódlapok (az eredeti 437-es karakter, illetve zárójelben a kód)

	CWI-2	852	CWI-1	SZKI	1250	Win 2
á	á (160)	á (160)	á (160)	á (160)	ß (225)	ß (225)
Á	Ć (143)	Á (181)	Ć (143)	ă (199)	- (193)	- (193)
é	é (130)	é (130)	é (130)	é (130)	Ú (233)	Ú (233)
É	É (144)	É (144)	É (144)	É (144)	+ (201)	+ (201)
í	í (161)	í (161)	í (161)	í (161)	Ý (237)	Ý (237)
Í	Ź (141)	Í (214)	î (140)	- (205)	- (205)	- (205)
ó	ó (162)	ó (162)	ó (162)	ó (162)	˘ (243)	˘ (243)
Ó	Ľ (149)	Ó (224)	Ľ (149)	Đ (209)	Ë (211)	Ë (211)
ö	ö (148)	ö (148)	ö (148)	ö (148)	÷ (246)	÷ (246)
Ö	Ö (153)	Ö (153)	Ö (153)	Ö (153)	Í (214)	Í (214)
ő	ô (147)	ő (139)	ô (147)	_ (219)	§ (245)	˘ (244)
Ő	ž (167)	Ő (138)	ž (167)	Ţ (221)	Ň (213)	ď (212)
ú	ú (163)	ú (163)	ú (163)	ú (163)	· (250)	· (250)
Ú	Ś (151)	Ú (233)	Ś (151)	Í (214)	+ (218)	+ (218)
ü	ü (129)	ü (129)	ü (129)	ü (129)	Ř (252)	Ř (252)
Ü	Ü (154)	Ü (154)	Ü (154)	Ü (154)	_ (220)	_ (220)
ű	ľ (150)	ű (251)	ľ (150)	_ (220)	ű (251)	ű (251)
Ű	ś (152)	Ű (235)	ś (152)	Ů (222)	_ (219)	_ (219)

A leggyakrabban használt DOS kódrendszerek (437, CWI, 852) összehasonlítása

DECIMÁLIS			128			144			160			176		
	HEXADECIMÁLIS		80			90			A0			B0		
		BINÁRIS	1000 0000			1001 0000			1010 0000			1011 0000		
			437	CWI	852	437	CWI	852	437	CWI	852	437	CWI	852
0	0	0000	Ç	Ç	Ç	É	É	É	á	á	á	░	░	□
1	1	0001	ü	ü	ü	æ	æ	Ĺ	í	í	í	▒	▒	□
2	2	0010	é	é	é	Æ	Æ	ĺ	ó	ó	ó	▓	▓	,
3	3	0011	â	â	â	ô	ő	ô	ú	ú	ú	│	│	□
4	4	0100	ä	ä	ä	ö	ö	ö	ñ	ñ	Ą	┤	┤	„
5	5	0101	à	à	ů	ò	Ó	Ľ	Ñ	Ñ	ą	╡	╡	Á
6	6	0110	å	å	ć	û	ű	ľ	ª	ª	Ž	╢	╢	Â
7	7	0111	ç	ç	ç	ù	ù	Ś	º	Ő	ž	╖	╖	Ě
8	8	1000	ê	ê	ł	ÿ	Ű	ś	¿	¿	Ę	╕	╕	Ş
9	9	1001	ë	ë	ë	Ö	Ö	Ö	⌐	⌐	ę	╣	╣	…
10	A	1010	è	è	Ő	Ü	Ü	Ü	¬	¬	¬	║	║	†
11	B	1011	ï	ï	ő	¢	¢	Ť	½	½	ź	╗	╗	‡
12	C	1100	î	Í	î	£	£	ť	¼	¼	Č	╝	╝	□
13	D	1101	ì	ì	Ź	¥	¥	Ł	¡	¡	ş	╜	╜	Ż
14	E	1110	Ä	Ä	Ä	₧	₧	×	«	«	«	╛	╛	ż
15	F	1111	Å	Á	Ć	ƒ	ƒ	č	»	»	»	┐	┐	‰

DECIMÁLIS			192			208			224			240		
	HEXADECIMÁLIS		C0			D0			E0			F0		
		BINÁRIS	1100 0000			1111 0000			1101 0000			1110 0000		
			437	CWI	852	437	CWI	852	437	CWI	852	437	CWI	852
0	0	0000	└	└	‹	╨	╨	đ	α	α	Ó	≡	≡	-
1	1	0001	┴	┴	□	╤	╤	Đ	ß	ß	ß	±	±	˝
2	2	0010	┬	┬	'	╥	╥	Ď	Γ	Γ	Ô	≥	≥	˛
3	3	0011	├	├	'	╙	╙	Ë	π	π	ń	≤	≤	ˇ
4	4	0100	─	─	"	╘	╘	ď	Σ	Σ	ń	⌠	⌠	˘
5	5	0101	┼	┼	"	╒	╒	Ň	σ	σ	ň	⌡	⌡	§
6	6	0110	╞	╞	Í	╓	╓	Š	µ	µ	÷	÷	÷	A
7	7	0111	╟	╟	ă	╫	╫	Î	τ	τ	š	≈	≈	¸
8	8	1000	╚	╚	•	╪	╪	ě	Φ	Φ	Ŕ	°	°	°
9	9	1001	╔	╔	–	┘	┘	©	Θ	Θ	Ú	∙	∙	¨
10	A	1010	╩	╩	—	┌	┌	®	Ω	Ω	ŕ	·	·	˙
11	B	1011	╦	╦	□	█	█	±	δ	δ	Ű	√	√	ű
12	C	1100	╠	╠	™	▄	▄	µ	∞	∞	ý	ⁿ	ⁿ	Ř
13	D	1101	═	═	›	▌	▌	Ţ	φ	φ	Ý	²	²	ř
14	E	1110	╬	╬	¦	▐	▐	Ů	ε	ε	ţ	■	■	·
15	F	1111	╧	╧	¤	▀	▀	¶	∩	∩	´			

ISBN 963-577-270-X

9 789635 772704